# UNPRINTABLE PHYSICS

## Introductory Physics for the Apple Computer

SHERIDAN SIMON

# Unprintable Physics

## Introductory Physics for the Apple Computer

# Unprintable Physics

## Introductory Physics for the Apple Computer

**SHERIDAN SIMON**
*Guilford College*

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

Editorial/production supervision:
Karen J. Clemments
Package design: Photo Plus Art
Package illustrator: Don Martinetti
Manufacturing buyer: Chris Johnston

Apple is a registered trademark of Apple Computer, Inc.

© 1984 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Library of Congress Catalog Card Number 83-062265

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-938259-3

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

To the physics majors of Guilford College, 1974-1982

C O N T E N T S


Preface . . . . . ix

Getting Started . . . . . 1

Mathematical Methods . . . . . 6

VECTORS . . . . . 6
GRAPH . . . . . 8
LEAST SQUARES FIT . . . . . 12
DEFINITE INTEGRAL . . . . . 13
NUMERIC/ANALYTIC . . . . . 15

Mechanics . . . . . 19

TRAJECTORY INQUISITION . . . . . 19
EXPEDITION TO IO . . . . . 20
ONE-D MOTION . . . . . 23
VISCOUS MOTION . . . . . 25
COLLISION . . . . . 27
MULTI-STAGE ROCKET . . . . . 28
DAMPED HARMONIC DEMO . . . . . 30
TWO-D HARMONIC MOTION . . . . . 32
MOMENT OF INERTIA . . . . . 34
KEPLER'S LAWS . . . . . 37

Thermodynamics . . . . . 39

MAXWELL-BOLTZMANN DISTRIBUTION . . . . . 39
P-V GRAPHS . . . . . 41
THERMODYNAMIC CHANGES . . . . . 43
THERMODYNAMIC CYCLES . . . . . 44
MOLECULAR MOTION . . . . . 46

Electromagnetism . . . . . 48

ELECTRIC FIELD . . . . . 48
CATHODE RAY TUBE . . . . . 49
UNIFORM B FIELD . . . . . 51
LRC CIRCUITS . . . . . 53
ELECTRICAL GENERATOR . . . . . 55

Wave Phenomena . . . . . 57

WAVES . . . . . 57
SUPERPOSITION . . . . . 59
SNELL'S LAW . . . . . 61
THIN LENS . . . . . 62

Modern Physics . . . . . 63

RADIOACTIVITY . . . . . 63
SPECIAL RELATIVITY . . . . . 66
HYDROGEN SPECTRUM . . . . . 68

# P R E F A C E

These programs are intended to help students learn physics more quickly, more easily, and in greater depth. They are not intended to replace either lectures or textbooks, but to act as a supplement to these more traditional teaching tools. They require no knowledge of programming on the part of either student or instructor. I have tried to design them in such a way that a student will need no more aid in their use and interpretation than is contained in this instruction pamphlet.

I have tried in the programs provided here to take advantage of what I see as a dynamic medium of communication, as opposed to the static medium of textbooks. The programs are of wide variety, but they have one thing in common: every one of them is capable of producing virtually unlimited different responses to student input.

Many of the programs involve graphs or sketches, where the student's inputs have direct, immediate, and visible effect on the output. This differs radically from a textbook presentation, where the number of such diagrams is strictly limited. Conceptually, use of most of these programs has much in common with experimental work. The student can keep all inputs but one constant over several runs of the program, changing that one and noting how it influences the results. Suggestions for utilizing this capability are made in the instructions for individual programs.

The programs can be used to advantage both by students having problems understanding basic concepts and by students seeking more challenging work. Since inputs are in the control of the student, they can be as simple - or as complicated - as the student desires.

While it was not possible for me to include all of the topics found in textbooks on a single disk, I have tried to provide programs that cover most of the major units included in a calculus-based physics survey intended for scientists and engineers. I have tried to place particular emphasis on areas that can be treated to greater advantage by computer than by textbook. Tables of constants, mathematical developments, detailed sketches and photographs, historical commentary, and detailed verbal discussion and analysis have no place here, and are not included.

Although it is somewhat difficult to generalize, I have tried to design a typical program with the idea in mind that an average student should spend one to two hours in mastering its contents. Rather than placing an added burden on the student, the time spent running these programs should mean less time spent reading and rereading texts, seeking the enlightenment provided by faculty and advanced students, and wandering about dormitory halls with a pained expression.

I am grateful to Dr. Paul Doherty of Oakland University, Rochester, Michigan, who very kindly donated his method for inputting functions. This is used in GRAPH, DEFINITE INTEGRAL, and ONE-D MOTION. The meticulous attention to detail of Paul as well as Dr. Larry Kirkpatrick of Montana State University in criticizing the programs has added substantially to their value.

I would like to thank Dr. Rose Simon for lending me her computer to write these programs, Dr. Hugh Van Horn for teaching me enough physics to write them, Dr. Rexford Adelberger for being super about getting me time off to write them, and Dr. Nancy Thompson for teaching me humility (unsuccessfully). I am particularly grateful to Doug Humphrey of Prentice-Hall, whose friends Vinnie and Rocko were an inspiration to me in getting this done on time.

# G E T T I N G  S T A R T E D

These programs are supposed to help students learn physics. In order to accomplish this, I've written them so they can be run by people with no knowledge of programming. However, you can't just walk up to the Apple and ask it for help on the homework for chapter six. This introduction will give you the instructions you need to get started. After that, each individual program has its own instructions in this booklet. If you already know something about Apples it won't take you long to read this section, but I advise you to do it anyway to make sure there is nothing surprising or unfamiliar going on.

Before I even get started, though, I want to give you an invitation. I want these programs, and the instructions, to be as useful as possible to first-year physics students. I still think this was the hardest course I ever took, and I want to be able to help students learn this material in the best way I can. For this reason, I want your comments and suggestions. Tell me what you like as well as what you don't like; I promise not to be insulted. The address is

Sheridan A. Simon
Physics Department
Guilford College
Greensboro, NC 27410

Formalities concluded, let's get on with the instructions. You should have available this instruction book, an Apple II computer with a TV monitor and a disk drive, and a disk. The disk is the square thing with the hole in the middle, a bit over five inches on a side. On the technical side, the Apple should operate under DOS 3.3 and have 48K of memory. This is true of most Apples, but if you aren't sure about this (which would not be surprising) ask your instructor.

Never run a computer by yourself before? Sit down in front of the Apple's keyboard, wipe the sweat off your hands, and calm the beating of your heart. Apart from exploding with sufficient force to demolish the entire Northern Hemisphere, which I assure you is very unlikely, the Apple is harmless, and even friendly. See? You've been sitting there thirty seconds already with no ill effects.

The computer programs you're going to run are recorded on a thing called a disk. A disk has much in common with a record. It spins around inside the disk drive, which acts like a high-speed turntable, and the different programs can be thought of as cuts on the disk. One difference between disks and records is that disks, like tapes, are magnetic. They can be erased and recorded over.

Your disk is sealed inside a black, square package, and you shouldn't open it. If you do, you'll ruin the disk and have to buy a new one. Disks are more delicate than records, and you have to treat them with respect. Handle them by their edges and don't leave them sitting near magnets, on stoves, or in places they'll be sat on. Since all the programs are on the disk, let's look at it. It's probably inside a paper sleeve of some kind.

Pull it out, black sealed package and all.

One side has a label on it, and that's important because that side must be up when you put the disk into the disk drive. You can see the disk inside the sealed package; be careful not to touch it. With the label side up you'll notice an oval opening in the sealed black package. The oval extends nearly from the hole in the center of the disk to nearly the edge. The edge that the oval nearly touches must be the first one into the disk drive.

Now open the door of the disk drive. This works like the latch on a suitcase, and flips up if you pull it a bit from the bottom. If there are two disk drives, open the door of the one labelled "Drive 1" or something similar.

Opening the door provides a horizontal slot just big enough for a disk. With the label-side up, turn the disk so the edge the oval nearly touches is right in front of the slot in the disk drive. Ready? Slide the disk into the disk drive. The label should be on the right side as the disk slides in. When the disk is entirely inside the disk drive, you'll hear a slight click and the disk won't slide any further. Now close the door of the disk drive.

The next step is to turn on the TV monitor. Since the monitor is nothing but a TV connected to the computer, the switch ought to be in the usual place and work the usual way.

Next, turn on the Apple. The switch is usually located on the left in the back. I don't think this makes sense either, but then I'm not a computer manufacturer. When you flip the switch you'll hear the disk drive spinning the disk around for a few seconds, then the computer will display a title page for a few seconds, then a copyright notice. The disk drive then spins again while more serious matters are loaded into the computer.

Once the disk drive stops spinning, you will be looking at something called the menu. The screen should show what's on the next page:

M E N U

SECTIONS:

1 MATHEMATICAL METHODS
2 MECHANICS
3 THERMODYNAMICS
4 ELECTROMAGNETISM
5 WAVE PHENOMENA
6 MODERN PHYSICS

TYPE A SECTION NUMBER TO SEE THE
PROGRAMS IT CONTAINS OR Q TO QUIT:

All this means pretty much what it says. If you hit 1, you're presented with a list of five programs having to do with mathematical methods in physics, for example. Below the list of programs is an instruction telling you to hit a program's number to make it run. The instructions for individual programs are in the rest of this booklet.

The things you type are called "input." What the computer responds with is "output." Sometimes something you input has to be followed by hitting the RETURN key. The instructions for individual programs tell you when this is necessary. Many of your inputs are numbers. In order to type these correctly, you need to know a few rules:

1. The zero is the odd-looking symbol to the right of the 9. It looks like this: Ø. Don't confuse it with the letter O.

2. Don't put commas in your numbers, ever. 1ØØØØ is right; 1Ø,ØØØ is wrong.

3. The scientific notation for the Apple works like that for many calculators. The number 1234 may be input as 1.234E3. The number .Ø1234 may be input as 1.234E-2. The E can be loosely translated as "times ten to the."

One thing that may be comforting is that you can't "hurt the computer" by typing something by mistake, no matter how dumb it is. Usually the computer can figure out what kind of mistake you made and tell you how to correct it. For example, if you are asked to input a mass and you type -18, the computer might say

THIS HAS GOT TO BE POSITIVE, FOOL.

and then ask you again to type the mass. Occasionally, you may make a more creative error, like typing & for the mass. The computer may respond with

and let you try again. In other programs I have told it to assume that you meant to type zero (Ø) if you hit something incomprehensible. This was necessary for technical reasons, so watch where you put your fingers. It doesn't cause anything disastrous to happen, but the results may be surprising. Finally, if you do something sufficiently weird that the computer can't figure out what to do next, it automatically returns to the menu so you can start over. None of these things causes any damage, and most of them happen only by accident. No hassle if you make a mistake once in a while; computers are very patient and won't tell the professor a thing.

You can quit any time you want except when the disk drive is spinning the disk. You can tell if this is happening if the red light below the door of the disk drive is lit (the label next to the light says "in use"). The drive never spins for long. Wait until it stops before you open the door, slide out your disk, reclose the door, and turn off the computer and monitor. You can even quit right in the middle of a program if you want to, and nothing bad happens. Just make sure the disk isn't spinning.

If the disk drive doesn't stop spinning within sixty seconds, something is messed up. Usually, you've forgotten to put the disk in the disk drive before turning the computer on. If this happens, push the top righthand button on the keyboard, RESET. This will stop the disk drive and let you start all over from the beginning.

The programs are designed to go along with your course. Each section includes programs on major areas (though not all areas) covered in first-year physics. You might look at the instructions for programs in the Mathematical Methods and Mechanics sections sometime soon, as you'll probably be using these first. Running a few of them is good practice, too, though you may not understand the output before you've covered the material either in class or in your textbook.

Since there are a total of thirty-two programs, you'll be running roughly one a week during a two-semester course. Some topics I've written programs for won't be covered in your course, though, and other programs can get used over and over in a lot of applications (GRAPH is one such).

Read through the instructions for each program once before running it, to familiarize yourself with what is going to happen. This only takes a couple of minutes and makes it a lot easier to use the programs.

The programs are supposed to be run several times each, changing the inputs. There are suggestions about this in the instructions for individual programs, but a key point is to have patience. Nobody learns physics in flashes of inspiration ("Aha! I'll bet $E=mc^2$!"). You've got to look at a program's output and see how it changes when you change input quantities. After a few tries, you'll gain some insight into the physics being described by the program. You should strive to understand the physical situation that the program represents. These programs are supposed to make that step easier, but they still only symbolize what is happening in the real world.

You can get a thorough understanding of each program's material in roughly one to two hours of work. You should actually save more time than

this, since you'll need less rereading of the textbook and such to gain a good grasp of a particular topic. Different people will need different amounts of time, however, and some programs are more complicated than others, so the time estimate is only a rough one. Let me know if they take you a lot less or a lot more, and why.

I wrote the instructions (and the programs) so you could have some fun with them while using them. There is no reason why physics has to be grim as well as difficult. But don't let this hide the fact that physics is a serious matter. The purpose of these programs is to help you to learn physics, and thereby to make a contribution in professional science or engineering. It is a subject filled with subtlety and complexity, but great beauty as well. My hope is that in seeing the patterns produced in various physical situations, by manipulating the inputs of the programs, you'll be able to appreciate sooner in your career the fascination physics can hold.

# V E C T O R S

This program has two sections. The first one allows you to input the magnitude and direction of a vector. It then displays the x- and y-components of the vector on a graph. The second section allows you to input the magnitudes and directions of two vectors. It then displays the sum of the two vectors on a graph.

You can use the first section of this program to see what happens to a vector's components as you change the vector's magnitude (keeping the direction constant) or its direction (keeping the magnitude constant). It can also serve to show you under what conditions a vector's x- or y-components are zero.

The second section can allow you to see what happens when different vectors are added together. The sum of two vectors depends not just on their magnitudes, but also on their directions, and you can see this by picking the magnitudes of two vectors and keeping them constant while trying different directions. Particularly important cases in physics are ones in which the angles between the two vectors are 0, 90, 180, and 270 degrees.

If you have hassles understanding vectors, start out by choosing the simplest numbers you can think of--like 1 for magnitudes and 0 or 90 degrees for directions. The computer doesn't mind your trying simple examples. I've found that they usually help in clarifying the situation a lot more than picking something weird to begin with (magnitude 2.718, direction 18.42 degrees, for example). With complicated numbers you have no idea what the results will be (at first). With simple ones you can more easily see how the computer's display makes sense.

When the program starts, you get to input 1 (for the components section), 2 (for the addition section), or M to return to the main menu. Hit 1, 2, or M (no RETURN is necessary).

If you hit 1, the computer draws a blank graph and asks you for the magnitude of your vector. The first time you run the program, I suggest you type a 1 for this (why get complicated?). Whatever number you choose to type, follow it by hitting RETURN. The computer then asks for the vector's direction, in degrees. For a first run, try 45. Once you've typed this number, hit RETURN. The computer then draws the vector, its x-component (on the horizontal axis), and its y-component (on the vertical axis). Below the graph, it prints the vector's magnitude, direction, and the sizes of its two components. If you use a magnitude of 1 and a direction of 45 degrees, both components come out to be .707. At the bottom of the screen will appear instructions on what to do next. If you hit C, the computer runs the example again for a vector of the same magnitude but direction changed by 10 degrees. This gives you an easy way to see what happens to the components as a vector is pointed in different directions.

The principal artistic failure of this section is that vectors sometimes look a bit like lightning flashes. The computer can draw great

horizontal and vertical lines but anything else looks a little ragged. Don't let this bother you. It's a lot better than drawing each one out as a scale diagram on graph paper as I had to do when I was learning this during the Napoleonic Wars.

If you hit a 2 when the program first starts the computer once again draws a blank graph and asks for the magnitude of vector #1. Type this, then hit RETURN. You are then asked to input the direction of vector #1, in degrees. Type this, then hit RETURN. The same steps (magnitude, then direction) get repeated for vector #2. After you've hit RETURN after typing the direction of #2 the computer draws a scale diagram of your two vectors and their sum. The first vector starts at the origin. The second starts at the end of the first vector. The sum extends from the origin to the end of the second vector.

If you followed my suggestion for a first attempt at the components section, try choosing the magnitudes of both vectors to be .707 and their directions to be 0 and 90 degrees. This shows you pretty clearly that a vector can always be thought of as the sum of its components.

Below the graph of the two vectors and their sum are printed the numerical values of the magnitudes and directions and instructions on what to do next. If you hit C, the direction of vector #2 is changed by 10 degrees, leaving everything else as it was before. This can allow you to experiment fairly easily with adding vectors that point in different directions.

# G R A P H

This program can draw the graph of any function you are likely to encounter in physics. You get to input the upper and lower limits on X and the function, F(X). The first thing the computer prints is

F(X) =

You get to type the function you desire to the right of the equal sign, then hit RETURN. In order to type the function correctly, you need to know something about the notation used by the computer.

The computer's notation for some of the common math operations is a bit different from what you're used to, but not a lot. For example, * is the symbol used for multiplication, so that 2 times X is written 2*X. Addition and subtraction are + and - as usual, but X divided by 3 is written X/3. As an example, suppose you wanted to see the graph of the function 7.5 X - .26. After the computer prints

F(X) =

you type

7.5*X-.26

and then hit RETURN.

For powers of X, the symbol ^ is used (over the N on the keyboard) so that the way to write X squared is X ^ 2 (X to the power 2). X cubed is X ^ 3, and X to the -.12345 is X ^ (-.12345). Use parentheses as you ordinarily would in algebra, but be a bit careful because there's only one kind--no brackets or braces.

Of the more complicated operations, some are the same as usual, like SIN(X), COS(X), TAN(X), and EXP(X) (which means "e to the x"). Trig functions have arguments in radians, not degrees. The symbol for natural log of X is LOG(X). Square root of X can be written as either X ^ .5 or as SQR(X). Incidently, the parentheses shown, like those around the X in SIN(X), are necessary.

For a good, horrible example let's try

$$\sqrt{1 - [\mathrm{COS}(X)]^2}$$

You put this into GRAPH as

F(X) = SQR(1 - (COS(X)) ^ 2)

Get the idea?

After you have input your function and hit RETURN, the screen clears, then LOWER LIMIT: appears near the bottom. Type the lowest value of X you are interested in seeing on your graph, then hit RETURN. You are then asked to input the upper limit on X, which is the highest value of X you want to see on your graph. Type this, then hit RETURN. The screen gradually turns white, starting at the left side and moving towards the right. The computer is scanning through your values of X, starting at the lower limit and going up to the upper limit, evaluating the function at 279 different values of X. If you want to see the graph of a complicated function, this may take several seconds, but it sure beats doing it yourself with graph paper and a calculator.

Once the computer finishes its calculations, it draws the graph of your function. You may see both the x- and y-axes in your picture, just one of them, or neither of them. It depends on the limits you set on x and the function you typed. For example, if your lower limit on x is 1 and the upper limit is 2, you won't see the y-axis (where x = 0) because it's to the left of what is shown in the picture. Below the graph are printed the lowest and highest values of both x and y that are displayed.

The last line on the screen asks you to type E for evaluation or Q to quit. If you type E, a small white dot appears on the line just drawn at the extreme left of the graph. Also, the information printed below the graph changes. Two lines tell you the values of the x and y coordinates at the white dot. The other lines tell you how to move the dot around on the graph (R moves it right, L left, and Q quits--real complicated). This is useful if you want to find approximate values of the roots of functions (the value of X where the function crosses the X-axis). A warning here is that the white dot moves in tiny jumps rather than continuously. Therefore, although the X and Y at the dot are shown to nine significant figures you probably won't get the dot to land exactly on the spot you want on the graph--it may be a bit too high or a bit too far left or something. For this reason you shouldn't trust values of maxima and minima, or roots, to better than 1% of the range of either X or Y shown on the graph. For example, if X runs from -1 to 1 the value of X shown by the dot should be trusted only to within about 1% of 2, or .02. If Y runs from -6 to -3, you can trust the value of Y shown by the dot to within about .03.

There is a help for this. Suppose you graph

```
F(X) = X^2 - 2
```

and want to know the function's positive root. You might first let X range from (say) 0 to 10. You can use the dot with the resulting graph to find that the root is within .1 of 1.4. Next, rerun the program with the same function but with more restrictive limits on X, say 1 to 2. This lets you find that the root is within .01 of 1.41. By continuing this process you can get as close as you want to the value of a root.

To speed up movement of the dot you can use the REPT key. If you want to move the dot a long way to the right, for example, press the R and the REPT (which stands for "repeat") and hold them down. The dot continues chugging to the right until you release the two keys.

If you hit Q to quit, the computer gives you the option of running the program again with the same function (hit A) or returning to the menu (hit M). The A option lets you change the upper and lower limits without retyping the function.

As usual, I suggest that you start out by using GRAPH on something simple, something you already understand. Once you are convinced the program makes sense, you can try it on more exotic functions. Eventually, you will get cocky and want to try it on a function that's undefined somewhere, like 1/X near X = 0. Go right ahead, but the results may be weird. This program is supposed to work on functions that have importance in physics--and nothing in physics is ever infinite.

Once you've figured out how to work this program, rejoice. It's about the most complicated one to run on the whole disk. The others may be harder to interpret, but most of them are easier to run.

Table 1

| Algebraic Notation | Computer Notation |
|---|---|
| $X + 1$ | X + 1 |
| $X - 1$ | X - 1 |
| $2 X$ | 2 * X |
| $X / 3$ | X / 3 |
| $X^{1.5}$ | X ∧ 1.5 |
| sin(X) | SIN(X) |
| cos(X) | COS(X) |
| tan(X) | TAN(X) |
| $e^X$ or exp(X) | EXP(X) |
| ln(X) | LOG(X) |
| $\sqrt{X}$ | SQR(X) or X ∧ .5 |
| $1.23 \times 10^{-4}$ | 1.23E-4 |
| $-2.3456 \times 10^{14}$ | -2.3456E14 |
| $\lvert X \rvert$ | ABS(X) |
| Random number between 0 and 1 | RND(1) |
| $\sqrt{(X / 2)^3 - 3 X^2 + \sin(4 X^2)}$ | SQR((X/2)∧3-3*X∧2+SIN(4*X∧2)) |
| -1 if X<0,<br>0 if X=0,<br>1 if X>0 | SGN(X) |

# L E A S T  S Q U A R E S  F I T

This program produces a graph of a set of data and calculates the slope and intercept of the straight line that comes closest to passing through all the data points. The program uses the method of least squares to accomplish this. This method chooses the slope and intercept of the line so as to minimize the sum of the squares of the distances in the vertical direction of all the data points from the line.

The program asks you to input the total number of data points you have. It will then ask you to input the X- and Y-coordinates of each one. Follow each of these numbers with RETURN. The X-coordinate is whichever variable you want to put on the horizontal axis; the Y-coordinate is the variable you want to put on the vertical axis.

I expect this program will be most useful to you in analyzing data from the lab, but it's a good idea to try it out first on something easy, like two points, (1, 1) and (2, 2), which lie on a line of slope 1, intercept 0.

Once you have input all the X- and Y-coordinates, the program will plot a graph showing your data points as + signs and including the least squares line. It prints below the graph the extent of the axes in both X and Y, so you can get an idea of the scale of the graph's axes. It also prints the slope and intercept of the least squares line, the straight line that comes closest to passing through all the data points. The last line invites you to hit D to see the data (the numbers you typed in for the coordinates) or Q to quit (no RETURN needed). You should always hit D to check the data. If I have to type in more than five pairs of data points, I'm almost sure to hit a wrong key on at least one of them, so I assume this happens to you, too. A wrong number puts a data point in the wrong place on the graph and messes up the least squares line, which tries to get close to a wrong point. If you have typed a wrong number, hit E to edit the data. The computer asks you which point you want to change (they are numbered). Type this, then hit RETURN. The computer then asks you for the new X- and Y-coordinates. After typing each of these, hit RETURN. The computer then replots the graph and recalculates the least squares fit.

You can always get from the data to the graph by hitting G or from the graph to the data by hitting D. No RETURN is needed in either case.

If you hit Q, you have the option of running the program again with a different set of data (hit A) or returning to the menu (hit M). No RETURN is needed for either of these.

# D E F I N I T E  I N T E G R A L

This program evaluates definite integrals. You get to choose the function you wish to integrate, the upper and lower limits, and something called the "number of intervals" that determines the accuracy of the computation.

The first thing the computer asks you for is the function F(X) that you wish to integrate. Defining this is the key problem, and fortunately it isn't too complicated. Most of the operations you might want to do have the same notation in computer language as in ordinary math. Check Table 1 at the end of the instructions for GRAPH for some examples.

As usual, it's a good idea to try this first on something you already know the answer to. A good choice might be to type just an X after the computer's F(X) =, then hit RETURN:

```
F(X) = X
```

The computer now prints

```
LOWER LIMIT:
```

Type the lower limit of your definite integral (this is the number near the bottom of the integral sign) followed by RETURN. You are then asked for the upper limit of the integral (the number near the top of the integral sign). Type this, then hit RETURN.

You now see

```
NUMBER OF INTERVALS?  (AN EVEN NUMBER
BETWEEN 4 and 1000):
```

This quantity is used by the computer in calculating the value of the definite integral. The bigger it is, the more accurate the value the computer calculates. However, the bigger it is, the longer the calculation will take. I suggest that you start with 100. Type this, then hit RETURN. The display on the screen changes to show you the function you typed, the upper and lower limits, and the computer's calculated value of the integral, along with instructions on what to do next. If you hit A to run again, you can try other limits or numbers of intervals for the same function as before without retyping it. In addition, you can keep the value of a previous input (like the lower limit) by hitting RETURN instead of retyping it. To try a new function you must first return to the menu.

Now run the program again with the same function, the same limits, and 200 intervals. If the computer's calculation for 200 intervals is close enough to that for 100 intervals, you're done. If your function is very complicated, or undergoes wild ups and downs between the upper and lower limits (you can check this by using GRAPH if you aren't sure), the 200-interval calculation may differ by more than you'd like from the 100-interval calculation. If so, try again with 300 intervals. Continue until two successive

calculations are so close together that you don't care about the difference (three significant figures is a common choice for "close enough").

Some definite integrals have infinite upper or lower limits. The computer can't deal with infinity (who can?) but you can often get decent estimates by using a limit that is large but not infinite. 100 is often big enough. To check, run again with the limit replaced by 200 and twice as many intervals. If the value is close enough to the last one, it's probably close enough to the true value.

If your function is complicated or you're using a lot of intervals, the calculation might take a while - up to several minutes for thousands of intervals (rarely necessary, fortunately). Until then, be patient.

The method of approximating definite integrals that is used by this program is called Simpson's Rule. It is one of the most commonly used methods, though not the simplest. I used it because it is a good compromise between speed and accuracy.

# N U M E R I C / A N A L Y T I C

This program allows you to compare results of numerical and analytic solutions of a particular problem, the damped harmonic oscillator.

Analytic solutions to equations of motion are algebraic formulas giving position as a function of time. They are obtained by rather complicated methods, and the resulting solutions are not always easy to interpret after they are obtained. Moreover, there are times when matters are worse. The solution of the equation of motion may only be expressible as an infinite series of complicated terms, or may not even be obtainable at all by any method you know.

An alternative to getting an analytic solution to an equation of motion is to solve the equation by numerical methods, one of which will be described below. These methods give solutions to equations of motion not as analytic expressions like cos(ωt) or exp(- r t) but as tables of numbers, like this:

| t | x |
|---|---|
| 0 | 0 |
| 1 | .1 |
| 2 | .4 |
| 3 | .9 |
| 4 | 1.6 |
| 5 | 2.5 |

This looks more like what one would obtain in a lab than anything else, and has both advantages and disadvantages compared to analytic solutions. The principal advantages of numerical solutions are that they give numbers directly and that a single basic method may be applied to an enormous variety of equations of motion. One computer program, like ONE-D MOTION in the Mechanics section, can solve an infinite number of different problems. In comparison, analytic solutions must always have numbers plugged into them before they can be compared with experiment, or to each other. Also, solving every new equation of motion (called ordinary differential equations by mathematicians) is a fairly nasty chore. Even though several useful methods exist for attacking equations of motion often encountered in physics, these methods, even with lots of practice, aren't as easy as simply running a program.

So why is a course in differential equations required for your major? Are we simply trying to keep the math department from having to start waiting tables for a living? Actually, no. Analytic solutions have advantages, too. One of the most important (and there are lots of others) is that analytic solutions are needed to check numerical ones. Computer programs are not trivial to write, and we always need some way of checking them to make sure they are working correctly. Even I used to have to do this, when I was merely human and didn't yet have a Ph.D. In addition, numerical solutions always contain errors.

Don't confuse errors with mistakes. A wrong command to the computer or a misplaced decimal point is a mistake, and can be avoided if care is taken. An error can't be avoided, like measurement errors in lab. We just hope to get them as small as possible. An error, as referred to here, will mean a difference between an analytic and a numerical solution to the same equation of motion after numbers are plugged into the analytic one. Now a warning from the Surgeon General:

NEITHER THE ANALYTIC NOR THE NUMERICAL SOLUTION IS THE "TRUE" SOLUTION. WHAT IS TRUE IS WHAT IS MEASURED IN THE LAB. ALL MATHEMATICAL SOLUTIONS OF ANY SORT ARE APPROXIMATIONS TO THE SITUATION IN THE REAL WORLD.

To obtain the numerical solutions shown by this program, I started by assuming that a damped harmonically oscillating object starts at rest at x = 1 when t = 0. This is like a mass attached to a bouncing spring at the top of its up-and-down motion. I take x = 0 to be the equilibrium point, where the mass will stand still if it is placed there at rest. I then solved the equation of motion

$$ma = F = - kx - bv$$

for the acceleration, a:

$$a = ( - kx - bv) / m$$

Suppose m, k, and b are known (they are inputs in the program, and in real life get measured in the lab). As I mentioned above, I assume that at t = 0 that x = 1 and v = 0. This means that the last equation can be used to find the acceleration at t = 0, since all the quantities on the right are known at this time.

I next used a trick derived from Newton's definition of the derivative. Since

$$dv/dt = a$$

it is possible to write an equation for the change in speed dv that takes place during the tiny interval of time dt right after t = 0:

$$dv = a\ dt$$

The value of v is known when t = 0, and I've got an equation for how much it changes during the small time dt. This means I can write an equation for the speed at the end of the time dt:

$$v(dt) = v(0) + dv$$

Flushed with victory, I tried to find the new position, x(dt), at the end of the time interval dt. Since

$$dx/dt = v$$

I got

$$dx = v\ dt$$

This means that the change in position, dx, during the time interval dt is v dt. The position at the end of dt is

$$x(dt) = x(0) + dx$$

That's looking good so far, but there is something a bit uncertain about the speed v in the equation for dx. Which one is it, the speed at t = 0 or the speed a little later at t = dt? If dt is really tiny, the two speeds will be nearly the same and it won't matter much which one we use. It will matter some, though, and this is the source of error in numerical calculations. Analytic methods avoid this by taking the limit as dt goes to zero.

"When in doubt, take the average" is a good rule in a lot of situations, and it's a good one here, too. In the program, I don't use either v(0) or v(dt), but their average, to determine x(dt). It's an approximation, but it gives small errors so long as dt is small. Physically, the smaller dt is, the less time the acceleration has to jiggle around the speed.

So let's see what the situation is. From x(0) and v(0), I can find a(0) by using

$$a(0) = (- k\ x(0) - b\ v(0)) / m$$

I can then use

$$v(dt) = v(0) + a(0)\ dt$$

and

$$x(dt) = x(0) + .5\ [v(dt) + v(0)]\ dt$$

to get the speed and position after dt has passed. Knowing the new v and x, I can get the value of the new acceleration:

$$a(dt) = (- k\ x(dt) - b\ v(dt)) / m$$

I can now repeat this process as many times as I want to get v and x at 2 dt, 3 dt, and so on. This is rather a pain by calculator (and you kids today don't know how rough it was in the old days, by cracky, with slide rules and candle light...), but such repetitive tasks are what computers were meant to do.

The program assumes x(0) = 1 and v(0) = 0 but lets you input m, k, and b. The mass must be 1-10 kg, the spring constant 1-100 newtons/meter, and the damping constant between 0 and 1. You also get to choose a value for dt, so you can see the errors (differences between the numerical and analytic solutions, and remember the Surgeon General's warning) that result. The smaller dt is, the better the two solutions agree. This is called "convergence." You can choose dt as small as 1/200 of the period, but no smaller. The only reason for this limitation is that the program takes too long to run if dt is too small. You have the option of having the computer automatically repeat the calculation with dt taken as half its previous value. This lets you run the program repeatedly for the same m, k,

and b so you can see the convergence of the analytic and numerical solutions more clearly as dt gets smaller.

The program displays a graph of both kinds of solutions with time on the horizontal axis. The numerical results are indicated by + and the analytic results by dots. The results are calculated for a total of three periods of the oscillator. If you choose dt too large, the numerical result may be a really poor approximation to the analytic one, possibly even right off the graph. If the computer tries to print a point off the screen, it will beep in pain to let you know. This is no cause for alarm, but you might try a smaller dt so both sets of points are at least on the graph.

# T R A J E C T O R Y  I N Q U I S I T I O N

This program produces a wide variety of trajectory problems so that you can test your understanding of related course material. Each time the program is run it produces a set of trajectory problems, of random types and with randomly-chosen data. You can choose to do 1, 2, 3, 4, or 5 problems; type this number, then hit RETURN. The problems are printed on the screen one at a time. Four questions are asked about each problem, also one at a time. Type your answer to each one in turn, following the number you type with RETURN. Each time you hit RETURN a new question will appear, until you're done with all the problems.

After you finish the four parts of each problem, the correct answers appear at the bottom of the screen. You can look at these as long as you want. Hit RETURN to go on to the next problem. At the conclusion, the computer displays your answers, the correct answers, and your score. Because of the possibility of roundoff error, your answer is considered correct by the computer if it is within .02 of the correct answer. This means .02 seconds for times, .02 meters for distances, .02 degrees for angles, and .02 m/s for speeds. For example, if the answer you give is 3.46 while the correct answer is 3.47, you get full credit, but if the correct answer is 3.43, you get no credit.

Alas, there is no partial credit, and the computer is not susceptible to bribes, threats, or excuses. On the other hand, there's no time limit - you get as long as you want to figure out each answer - and there's nothing stopping you from quitting any time you want. Note also that if a particular problem is too easy (or too hard) you can go on to the next one by typing 0 for all four answers. It hurts your score, but no one but you is keeping track of that anyway.

One peculiarity you will notice immediately is that the questions have words broken at the right edge of the screen and continued at the left edge on the next line. This is done without hyphens and without any attention to standard usage in breaking words at the ends of lines. This unfortunate situation is necessary to squeeze all the lines of printing on the screen at once. It's not all that much of a problem, but I thought I should mention it lest you think there is something wrong with your screen.

Even if you're pretty good at these, it's rather hard to get 100%, and not many people manage to.

# E X P E D I T I O N  T O  I O

The first American expedition to Io is in trouble.

Io is the innermost of Jupiter's four giant moons. NASA has sent a manned expedition there to survey the moon for new supplies of raw materials. Six of the seven astronauts on the Io mission left the expedition's landing craft soon after touchdown. They travelled in a Rover, an airtight vehicle designed for the rugged terrain and low gravity of Io. When they were several kilometers from the lander, an unexpected volcanic eruption began. The crust of Io is thin, and the stresses induced by the eruption caused collapses, forming deep crevasses. Gases emitted from the volcano pour across the surface, limiting the visibility of those on the Rover to a few meters.

Because of the lack of visibility, the Rover crew is unaware of the dangerous crevasses seen by the infrared detectors of a satellite left in orbit for mapping purposes. As the only expedition member left on board the landing craft, you receive the satellite's pictures and immediately try to contact the Rover and your six friends - only to find that their radio has stopped working.

The Rover, with your friends inside, can clearly be seen in the satellite picture that appears on your screen. They are wandering aimlessly, nearly blind, attempting to find their way back to the landing craft before their air is exhausted. Unaware of the deep crevasses opened in the crust by the volcano, they might topple into one at any moment.

You see just one chance. The satellite can send you the position of the endangered Rover. You have available a spring-loaded device meant for launching explosives for seismic experiments. It can launch fairly heavy loads, and can be set at any desired angle both horizontally and vertically. Unfortunately, it can launch loads only at one speed: 100 meters per second. Your hope is to load some spare air tanks in the launcher and have it drop them close enough to the Rover that your friends can spot them when the tanks land.

You are able to control the direction in which the loads are launched by inputting two numbers from the keyboard. The first is an angle called phi. It is measured in such a way that East is phi=0, North is phi=90, West is phi=180, and South is phi=270 degrees.

Since the satellite gives you the position of the Rover in terms of meters east and meters north of the landing craft, you have to use your knowledge of vector components to calculate the correct value of phi.

The second input number is called theta. This tells the launcher at what angle to the ground the load is to be launched. Horizontal is theta=0, vertical is theta=90 degrees (which would drop the load on your own head). Theta must be calculated from the distance to the Rover (vector components again) and the speed at which the loads are launched (100 m/s). To do this, you must use your knowledge of trajectory problems.

The infrared picture you receive from the satellite has low resolution, as you will see. The lander will appear as a square in the lower left corner, the Rover as a black dot. It will move from time to time in a random direction as your friends try desperately to find their way back to the lander through clouds of volcanic gases. Most of Io's surface is red in color, possibly from sulfur or phosphorus compounds, but some recent lava flows have left white areas, possibly made of ice (Jupiter and its moons are more than five times as far from the sun as the earth). The crevasses opened by the volcano (northwest of the lander) are shown as black.

You have enough air tanks to try three launches in your attempt at rescue. There is always the danger of the Rover's wanderings dropping it into a crevasse, or of it running out of air, so you need to work fast - and accurately.

A vital piece of information: the acceleration of gravity on Io is 1.49 meters per second squared.

The program begins by putting on your screen the picture transmitted from the satellite, which takes a minute or so. It then asks you to input a level of difficulty, a number 1-10. This has to do with how long a time you will have between the Rover's position shifts. Every time it moves, you have to recalculate phi and theta to launch the air tanks in the correct direction. 1 is easiest (longest time), 10 is hardest (shortest time). Hitting 1 also gives a longer time before the Rover runs out of air. You are warned, however, that times in the program are fairly random. Even at level 1 there are sometimes only a few seconds before the Rover moves, though often it stands still for a minute or more.

Once you choose a level of difficulty, type it, then hit RETURN. The computer gives you the coordinates of the Rover in meters east and north of the lander and asks you to input phi (in degrees). Type this number, then hit RETURN. The computer then asks you to input theta, also in degrees. Type this, then hit RETURN. If your load of air tanks lands close enough to the Rover to be retrieved, you will be told. If not, the landing place will flash so you can see how far off you were.

Just to add to the fun, I have included two further difficulties. One is pretty easy to understand, just frustrating. The Rover may be out of range of the lander, so that no matter what you choose for theta (even 45 degrees) the air tanks can't reach your friends. If you luck out, they may wander into range before they run out of air. The second difficulty is so fiendish I was smiling for days after I thought it up. When you are entering phi and theta, the computer will sometimes not receive your input number. When you hit a key, make sure the number (or decimal point) you type actually appears on the screen. If it does, you're OK. If it doesn't, type it again. This is a purely random effect, so don't hit the key harder or anything - just type it again.

The purpose of this program is to aid you in learning to think and calculate numbers under high pressure situations. Although this category includes tests you take in school, there are lots of situations in professional work after graduation in which you are operating under constraints of time or conditions as well. In a sense, I tried to make this simulation as "unfair" as possible: you get no partial credit, time is limited, you don't

know how much time you have, the problem keeps changing, and even if you type the right answer quickly enough you might make a typing mistake.

I'm not going to suggest a winning approach in detail, but you should have a calculator handy, and it helps to think through your strategy before starting the program. If you are successful at level 10, my heartiest congratulations.

# O N E - D  M O T I O N

This program allows you to give the computer a formula for the acceleration an object feels and to then study the resulting motion. The motion is displayed on two graphs, one of position and time and the other of velocity and time.

When you run the program, the first thing the computer asks for is the formula for acceleration. It prints

A =

To the right of the equal sign you might type a 0 (for no acceleration), a constant such as -9.81, or something more complicated, like -X. For still more interesting (i.e. complicated) possibilities, you should refer to Table 1 following the instructions for the program GRAPH. This table contains many of the computer language versions of common functions and operations, like addition, multiplication, trig functions, and so forth. The acceleration can depend on position (X), speed (V), time (T), or any combination of these. After you type the formula for acceleration, hit RETURN.

Let's suppose that you want to study the motion of freely falling bodies near the surface of the earth in the absence of air resistance. After the computer prints A =, you type -9.81, then hit RETURN.

The program now asks you to input a time step. The two graphs (position-time and velocity-time) that the computer displays show X and V at twenty different times, each separated from the last by an interval called the time step. You get to choose this. If your choice is 1, the computer will show you what happens over a total time of 20 seconds--a very long time for a freely falling body. Sensible values of the time step depend on the nature of the acceleration.

Once you've input a value for the time step, and hit RETURN, you are asked for the values of initial position and initial velocity. After typing each number, hit RETURN. The computer then plots the position-time and velocity-time graphs and prints a few pieces of useful information below them, along with instructions on what to do next. If you hit A to run again, the computer keeps the same formula for acceleration and lets you try new values of time step, initial position, and initial speed. If you want to keep your previous choice for any of these three inputs, hit RETURN instead of retyping it. If you want to try a new formula for acceleration, you have to go back to the menu first (hit M).

With this program it is particularly important to start simple. A good first choice is acceleration equal to -9.81, initial position 100 meters, initial speed 0. This simulates an object being dropped from a height of 100 meters. Make sure that the graphs make sense to you using various time steps (.01, .1, and 1 second, say). Remember that as a general rule if you can't understand the simple situations you'll never understand more complicated ones. Even if you understand some of what's going on you could miss

subtle points that may be interesting, important, or both.

Sometimes you may give the computer orders that lead to results so weird the program can't handle them. This almost always means you've tried to do something non-physical, like letting an object fall for a billion seconds with an acceleration of -9.81 meters per second squared. In any case, the result will be a beep and an error message that tells you there's something wrong with your formula for acceleration. This is no major problem. Figure out why what you tried was stupid (happens to the best of us--though not me, of course) and try again. The most common problem is position or speed getting so big the computer can't handle the numbers. The usual remedy is a shorter time step.

For the overconfident, foolhardy, or expert student there are complications that can be added. If you wish, for example, you can make acceleration a function of speed, V, as well as position, X. Such accelerations exist; air resistance is one example. But don't get carried away. Most of the possibilities lead to non-physical output. One that does lead to physically reasonable results is

```
A = -9.81-SGN(V)*.01*V^2
```

(see Table 1 at the end of the instructions for the program GRAPH if you don't understand the symbols). Choose initial position to be big, like 1000, and initial speed to be 0. The results are similar to what happens to a skydiver with surface area about 1 square meter and mass about 100 kilograms. A skydiver with more surface area has the constant out front bigger than .01; one with a smaller mass also has a bigger constant.

The acceleration can be made a function of the time, T, as well. I advise you to leave out any mention of T in your acceleration formula until you are satisfied that you clearly understand the results for functions of position, X, and speed, V.

To sum up, the acceleration can be a constant, a function of X, a function of V, a function of T, or some horrible combination. The more complicated the combination is, the less likely it is to make physical sense and the less likely it is that you can understand the results, so START SIMPLE.

# V I S C O U S  M O T I O N

This program demonstrates the settling of small particles in a stream. The particles may be natural, such as pollen or volcanic ash, or artificial, like industrial waste. The particles are carried in the horizontal direction (downstream) by the moving water. The motion of a particle in the vertical direction is controlled by three forces. These are gravity, buoyancy, and the viscous force caused by the interaction of the particle with the water as the particle drifts vertically. I'm assuming for simplicity that all of the particles are spherical. The relevant formulas are

$$F(\text{gravity}) = - mg$$

$$F(\text{buoyancy}) = \rho V g$$

$$F(\text{viscous}) = -6\pi r v \eta$$

Here m is the particle's mass, g the acceleration of gravity (9.81 meters per second squared), $\rho$ the density of the water, V the particle's volume, r the particle radius, v its velocity in the vertical direction, and $\eta$ is the viscosity of the water.

Gravity always points downward, hence the negative sign in the formula for gravitational force. The buoyancy is due to the water displaced by the particle, and always points upward. The viscous force has a direction opposite to that of v, the velocity of the particle in the vertical direction. If the particle is moving downward, the viscous force pushes up on it, and vice versa. This is a lot like friction. The formula for viscous force is called Stokes' law.

The program allows you to input a total of six quantities, four of them properties of the stream and two of them properties of the particles themselves. The stream properties are temperature (hotter water is less viscous) and the depth, length, and speed of the stream, which affect the output numbers and the picture. The width of your screen in the horizontal direction represents the length of the stream and the height of the picture in the vertical direction represents its depth. A running display of elapsed time in days (D), hours (H), minutes (M), and seconds (S), and distance travelled downstream by the particles, in kilometers, is included to show the stream's speed.

The two properties of the particles that you get to input are their specific gravity (density divided by the density of water, 1000 kilograms per cubic meter) and their average radius, in microns (1 micron = $10^{-6}$ meters). For rough estimates, rocks often have specific gravities of 2 to 3. Metals are more like 5 to 10, or even more for dense materials such as lead, tin, and plutonium. To get an idea of good radii, one micron is really, really fine dust. 100 microns is just visible. Since collections of particles are rarely all of the same radius, the program uses your input as an average. The radii it computes motions for are a random distribution with an average equal to your input.

Notice that your choice of specific gravity will affect the particle mass in the equation for F(gravity) as well as the density of F(buoyancy). Your choice of radius influences the mass in the equation for F(gravity), the volume in the equation for F(buoyancy), and also appears in the equation for F(viscous).

You should treat this as an experiment to get an idea of the physics. Keep all the inputs constant except one, and run the program repeatedly with different choices of that one until you understand the results. Then repeat the process with another input variable. If you hit A to run again after each experiment, you can keep the previous value of any of your inputs by hitting RETURN instead of retyping it.

In the picture, the 100 particles are shown as black dots. They are dropped into the stream from the upper left. The stream is the shaded area occupying most of the picture. The particles will drift to the right with the moving water as time goes on. The white area at the bottom is the bottom of the stream, soft sand, old beer cans, and so forth.

One peculiarity affects the picture. When a particle hits the bottom, it stops moving and the computer no longer calculates new positions for it. This means that it computes the positions of the remaining particles a bit faster than before. If a lot of the particles drop to the bottom the motion of the remaining ones to the right may seem faster than before. This is not really true, and the clock at the bottom of the picture can always tell you the correct value of elapsed time.

# C O L L I S I O N

This program allows you to watch two objects collide in two different reference frames. One of these is called the LAB frame. In this reference frame, one of the two objects starts out at rest and the other comes along and hits it. The observer (that's you) is sitting at rest and watching. The second reference frame is called the CM, or center of momentum frame. In this one, the observer (that's you again) is moving in such a way that both objects, even the one standing still in the LAB frame, seem to move. This only SOUNDS weird. Move your head from side to side. Don't things appear to move? That's all the CM frame is doing. The speed of the observer is carefully calculated in the program so that the total momentum of the two objects in the CM frame is zero. There is only one particular speed for which this is true.

The purpose of this program is to let you see what happens in a collision in both reference frames. I had a horrible time understanding this as a freshman, despite a rather patient professor. My motivation in writing the program was that I think if I'd had it to use this mess would have been a lot clearer to me.

You get to input the masses of both objects and the initial speed of the moving one in the LAB frame. You should note that the heavier one moves more slowly in the CM frame. Since momentum is mass times velocity, the more massive object needs less velocity to cancel out the momentum of the less massive object.

You also get to input the percentage of kinetic energy that is lost in the collision in the CM frame. If the two objects are very rigid, like steel, almost no kinetic energy is lost and the collision is called "elastic." If the two objects are made out of something softer, say dough, almost all of the kinetic energy present in the CM frame will be lost. This is called a "completely inelastic" collision. The lost kinetic energy gets used up in distorting the shapes of the two objects, producing heat, and making "clunk" sounds.

As usual, treat this like an experiment. Start simple - assume equal masses and either 0% or 100% of kinetic energy to be lost in the collision. If the results make sense to you, try changing one of the inputs a bit and run again. The results will be different; make sure you understand why they changed the way they did.

If you run the program again by hitting A, you can keep your previous choice for any input by hitting RETURN instead of retyping it.

# M U L T I - S T A G E  R O C K E T

This program allows you to design and test fly a simulation of a multi-stage rocket. While not as sophisticated as NASA would require, the program includes much of the basic physics of rocket flight, the most severe restriction being the neglect of air resistance.

Two major forces act on these simulated rockets, gravity and rocket thrust. The relevant formulas are

$$F(\text{gravity}) = -mg$$

$$F(\text{thrust}) = - v_{ex}\, dm/dt$$

The first formula should be a familiar one. The minus sign is present because gravity always acts in a downward direction. The second force is due to the momentum transferred to the rocket as it expels hot gases downward. The speed of these gases relative to the rocket is $v_{ex}$ (always positive). The mass of the rocket is symbolized by m. As fuel is burnt and expelled, the mass of the rocket decreases. The rate of change of the rocket's mass, dm/dt, is therefore always negative. Taking into account the minus sign leading the two factors in the equation for F(thrust), this force can be seen to be always positive, or upward. The net force acting on the rocket is F(gravity) + F(thrust). Unless F(thrust) is big enough to make the sum of the forces positive (upward), the rocket will stay on the ground.

A vital point here is that the weight of the rocket steadily decreases as it burns and expels fuel. The mass m of the rocket gets smaller as fuel is exhausted, so F(gravity) gets smaller, too.

The program will allow up to four stages to be simulated. The bottom stage is stage #1. It is the first one ignited, and when its fuel is exhausted it is dropped by the remaining stages. It is dropped because it would take the energy of burning fuel to push it higher, and it is no longer useful once its fuel is gone. The second stage then ignites (usually) and propels the rocket higher and faster. The process repeats for each successive stage.

Your goal is to design a rocket capable of pushing its final stage to escape speed, a little over 11 kilometers per second. An object moving this fast will never return to the earth. Real launch vehicles for spacecraft have to cope with air resistance, turning motions to place payloads in desired orbits, erratic performance of rocket engines, and other horrible problems, but this program gives you the basic idea. It is a good idea to have some paper handy to write down each set of inputs you try and the results, since for multi-stage rockets you have a lot of inputs to consider.

The first input you choose is the number of stages. The first few times you run the program, try one stage. After you type this number, hit RETURN. One stage won't reach escape speed (probably) but this simple situation allows you to get some idea of how rockets perform.

The program next asks you for the mass of each stage (in tons) in turn. Choose these to be anything you like, but if you choose something too outrageously big or small the computer may make a nasty comment about your intelligence or ancestry. The computer assumes that 90% of the mass of each stage is fuel. After you type the mass of each stage, hit RETURN.

Next, the computer asks for exhaust speed. This is the $v_{ex}$ in the formula for F(thrust). The bigger it is, the harder the rocket will be pushed upward. Typical speeds are around 2500 meters per second. This simulation assumes the same exhaust speed for all stages and that it remains constant as each stage burns its fuel.

Finally, the program asks for the burn rate of the fuel in each stage in tons per second. This is the absolute value of the factor dm/dt that appears in the equation for F(thrust). The bigger this is, the bigger the thrust, but the faster the fuel is consumed. You should try different values to see what the best approach is. Hit RETURN after typing each burn rate.

The program plots graphs of speed and height versus time and prints a running summary of important data as the simulation proceeds. If the rocket burns for too long, points no longer appear on the graphs but the running summary continues. Study the graphs and other results with one-stage rockets for various choices of mass, exhaust speed, and burn rate before moving on to the more complicated multi-stage case. The changing mass of the rocket is a particularly important factor in understanding its behavior.

If you hit A to run again, you can keep the previous value of any input by hitting RETURN instead of retyping it.

# D A M P E D  H A R M O N I C  D E M O

This program demonstrates damped harmonic motion. Although it can be interpreted more generally, in writing it I pictured a mass bouncing up and down on the end of a spring. You get to input the mass (m) in kilograms, the spring constant (k) in newtons per meter, and the damping constant (b) in kilograms per second (the units of b aren't easy to make sense of). The mass and spring constant are most important in determining the period of oscillation, while b mostly influences how fast the oscillation "damps out"--gets smaller and smaller as the oscillator loses energy to its surroundings. You also get to input the total time to be displayed on the graph of position versus time that the computer will plot. After typing each input, hit RETURN.

The program uses the formula

$$x(t) = \exp(-bt / 2m) \cos(\sqrt{k / m - b^2 / 4m^2}\, t)$$

to plot the position of the mass at the end of the spring as a function of time. Despite its complicated looks, it is not the most general possible formula for this kind of motion, a matter I found a considerable source of depression as a freshman. The idea of this program is to allow you to see a graphical representation of a rather complicated formula to allow you to get some insight into what it means physically. To help this, I've also included a friendly little dot (the mass) that bobs up and down in much the same way the mass on the end of a real spring would for your choices of m, k, and b.

The program prints 200 points, equally spaced over the total time you input. A "clock" below the graph gives a running measurement of the time as it passes and more points are plotted. This does not measure the same time as your watch. It is speeded up or slowed down depending on what you have chosen for total time displayed on the graph.

Notice in the formula for x(t) that the cosine term contains a square root. If the quantity

$$k / m - b^2 / 4m^2 < 0$$

the square root is imaginary. Physically, this happens if b is too big--a lot of damping, like trying to bounce the spring in a vat of molasses. If damping is too big, the mass doesn't oscillate--it just sort of sags back down to equilibrium. The motion in this case is called "overdamped." If

$$k / m - b^2 / 4m^2 = 0$$

(which rarely happens) the motion is called "critically damped." Overdamped or critically damped motion requires a different formula from the one I included with this program. If you pick too big a value for b, the computer will print a warning and let you start over.

Start out by choosing b=0. This lets you try various values of k and m to see what they do to the period of oscillation. They try different

values of b to see what happens when the motion is damped. It may help you to know that typical values of k range from about 1 (very loose spring) to 100 (very stiff spring). If you run again by hitting A, you can keep the previous choice for any input by hitting RETURN instead of retyping it.

One warning, about total time. If you choose this to be too big, the graph can look very strange. If you can't understand the results, try running again with a shorter total time.

# T W O - D  H A R M O N I C  M O T I O N

This program demonstrates what happens when an object experiences harmonic oscillations in both the x- and y-directions at the same time. It may be useful to you to run a simpler program, WAVES in the Wave Phenomena section, before running this one.

Motion in the two directions is given by

$$x(t) = A_x \cos(\omega t + \phi_x)$$

$$y(t) = A_y \cos(n\omega t + \phi_y)$$

$A_x$ is called the "amplitude in the x-direction." It tells you how big the horizontal oscillations are. If $A_x$ is big, the object swings rather widely in the x-direction. $A_y$, the amplitude in the y-direction, does the same thing for the other component of the motion. $\phi_x$, called the "phase" of the x-direction motion, tells you what the x-component of the object's position is at t=0. $\phi_y$, tells you where on the y-axis the motion starts. $\omega$ is called the "angular frequency." It affects how fast the object moves in each direction. Although you can't adjust $\omega$, you can choose n, the "frequency ratio," which determines how many oscillations occur in the y-direction for each one in the x-direction.

The output of the program is two graphs and a sketch. It takes the computer roughly thirty seconds to draw them. The graphs show x versus t and y versus t. The sketch shows a dot tracing out the path in x and y defined by your inputs. The picture is arranged so that the x-axis in the x-t graph runs parallel to the x-axis in the sketch, and the y-axis of the y-t graph is parallel to the y-axis in the sketch.

An enormous number of patterns can be followed by objects obeying equations of motion like those above, and there are five inputs under your control: the two amplitudes, the two phases, and the frequency ratio. A good place to start here is to input zero for both $A_x$ and $\phi_x$ and 1 for the frequency ratio (hit RETURN after you type each input). If $A_x$ is zero, the object won't move at all in the x-direction. This allows you to run the program repeatedly with various values of $A_y$ and $\phi_y$ until you understand what both of them do to motion in the y-direction. You should then input zero for both $A_y$ and $\phi_y$ and 1 for the frequency ratio and run the program a few times with various choices for $A_x$ and $\phi_x$ to see what they do to motion in the x-direction. Once you feel comfortable with these results (which are undamped harmonic motion in ONE dimension), you can try non-zero values of both $A_x$ and $A_y$ while keeping the frequency ratio at 1. Possible motions for various choices of the amplitudes and phases include straight lines at all angles, ellipses, and circles. You should not only try to produce these, but should strive to see WHY they are produced.

Finally, you can experiment with changes in the frequency ratio. This can produce some very strange-looking patterns called Lissajous figures.

They aren't as familiar as the simpler patterns produced for a frequency ratio of 1, but you can understand them fairly easily by watching the display and they are quite important in some advanced applications.

If you hit A to run again, you can keep your previous value of any input by hitting RETURN instead of retyping it.

# M O M E N T  O F  I N E R T I A

This program allows you to find the moment of inertia of a point, line, ring, or any combination of points, lines, and rings. The idea of the program is to allow you to get some intuition for the rather complicated concept of moment of inertia by seeing what its value is in various circumstances.

The program assumes that the object or objects you input rotate about an axis perpendicular to the screen and passing through a small + at its center. The + is at (0, 0) in the coordinates used in the program.

When you run the program, the first thing you see is a blank picture with the + at the center and a question printed below:

ADD AN OBJECT (A) OR QUIT (Q):

If you type a Q (no RETURN needed here), the computer calculates the moment of inertia of all the objects in the picture. Since there aren't any yet, this comes out to be zero, which is comforting but not very interesting. If instead you type A (no RETURN needed) you see

POINT (P), LINE (L), OR RING (R):

Type P (no RETURN needed). You are then asked for the x-coordinate of the point (type it, then hit RETURN), the y-coordinate (type it, then hit RETURN) and the point's mass. I know that points in the mathematical sense don't have mass. A "point" here is just an approximation for a mass that is very small in size compared to its distance from the rotation axis passing through the +. Type a mass, then hit RETURN. The computer then puts a dot on the screen at the point whose coordinates you input. You now see again the invitation to add an object or quit.

Let's add another object. Hit A. This time, we'll do a line, so when the next question appears type an L. You now see

VERTICAL (V) OR HORIZONTAL (H):

Type V for this, then hit RETURN. You now see

X - COORDINATE OF LINE:

Type a value for this, then hit RETURN. You now see

Y-COORDINATE OF TOP OF LINE:

Type a value for this, then hit RETURN. You now see

Y-COORDINATE OF BOTTOM OF LINE:

Type a value for this, then hit RETURN. You now see

MASS OF LINE:

Type a mass for this. Note that a "line" here is just an approximation for an object very narrow compared to its length, like a meter stick. Once you've typed the mass, and hit RETURN, the computer draws your line in the place you told it to, and once again prints.

ADD AN OBJECT (A) OR QUIT (Q):

This time, let's add a ring. Hit A, then R (no RETURN needed). You see

X-COORDINATE OF CENTER:

Type a value for this, then hit RETURN. As you might have guessed, you now see

Y-COORDINATE OF CENTER

Type a value, then hit RETURN. You are now asked for the ring's radius. Type this, then hit RETURN. You are next asked for the ring's mass. Type this, then hit RETURN. The computer obediently draws a ring with center and radius as you ordered, and once again asks if you want to add an object or quit.

By now you've probably typed at least one number that the computer objected to. In addition to obvious ones like negative masses, the computer wants all x-coordinates between -10 and 10 and all y-coordinates between -5 and 5. It will draw rings that extend outside this region, but their centers have to lie inside it.

Hit Q to see what you've done. The computer prints the moment of inertia of the complicated mass of objects you told it to draw. It's the right value, I assure you (all masses in kilograms, all distances, lengths, and radii in meters, all moments of inertia in kilogram-meters squared). However, the value of the moment of inertia isn't easy to interpret because the situation is too complicated. This exercise was only meant to show you the mechanics of how to run the program.

To use it to learn something about the physical quantity called moment of inertia, you've got to start simple. I suggest trying a single point to begin with. Run the program a few times with various masses and positions for the point. By finding the moment of inertia each time you rapidly gain some insight into how mass and distance from the rotation axis both influence the moment of inertia. Basically, this quantity tells you how hard it is to start the object or objects in the picture rotating about the +. If they are far from the +, or massive, they have a lot of moment of inertia.

This program is also useful in checking to see if you understand the formulas for calculating moment of inertia. You probably have available formulas for the moment of inertia of a ring about its center and a rod about its center, or about one end. You might see if you can demonstrate that they agree with the values the program gives. You may also have seen the Parallel

Axis Theorem. The program can allow you to experiment with this quite nicely.

If you're patient, you can construct quite elaborate and sometimes useful combinations of shapes. A large number of lines can APPROXIMATE a solid rectangle. A large number of rings can APPROXIMATE a solid circle. However, learning how to derive the relevant formulas is probably better than trying to use the program for purposes for which it wasn't really intended.

# K E P L E R ' S  L A W S

Kepler's Laws are good approximations to the behavior of planetary orbits, but can be somewhat confusing if all you have available as examples are diagrams in a book. This program allows you to produce examples of all three laws for a wide variety of possible orbits.

The first thing you see when running the program from the menu is an invitation to choose which law you want to see an example of. Type 1, 2, or 3. No RETURN is needed.

The First Law is a statement that planetary orbits are ellipses with the sun at one focus of the ellipse. This section of the program asks you first to input the semi-major axis of the planet's orbit in units called A.U. The semi-major axis is half the distance across the widest part of the elipse. Its symbol is A. Since for planets this comes to a lot of meters, astronomers often use distances measured in units of the average distance of earth from the sun. This number is known to a startlingly large number of significant figures, but is roughly 1.495 E11 meters. This distance is defined to be 1 Astronomical Unit, or A.U. Semi-major axes for the real planets range from just under .4 A.U. for Mercury to nearly 40 A.U. for Pluto. You can find these numbers in most astronomy books if you're interested. The program doesn't require you to use the numbers for the real planets.

Type the semi-major axis (in A.U.) you want to see, then hit RETURN. You are now asked to input the orbit's eccentricity. This ranges from 0 (if the orbit is a perfect circle) to nearly 1 for a very long, narrow ellipse. The biggest eccentricity of any planetary orbit is about .25 for Pluto, but comets often have highly eccentric orbits, and the program works for these, too. Once you type an eccentricity (and hit RETURN) the computer will draw the orbit, put a dot representing the sun at one focus, and print some data below the picture. The data includes your input values of semi-major axis and eccentricity plus your planet's orbital period in years (earth's, shockingly, is 1 year) and the planet's perihelion distance. Perihelion is the closest the planet gets to the sun. For circular orbits (eccentricity=0) this is the same as the semi-major axis, but for orbits with eccentricities near 1 the perihelion can be very small compared to the semi-major axis.

The Second Law states that the radius vector from the sun to a planet sweeps out equal areas in equal times. This means that every one-tenth of a planet's orbital period the line connecting the planet and the sun will cover one-tenth of the total area of the planet's elliptical orbit. When the planet is close to the sun this line is short, but the planet is moving quickly in its orbit. On the other hand, when the planet is far from the sun the radius vector is long, but the planet doesn't move along in its orbit very fast.

As in the case of the First Law, the program asks you to input the planet's semi-major axis (in A.U.) and eccentricity, each number followed

by RETURN. The computer then prints a picture of the orbit divided into ten sectors, alternating black and white. The areas of all sectors are equal (as closely as the computer can draw them) which means that the planet takes one-tenth of its total orbital period to traverse the part of its orbit covered by each sector. If the orbit has high eccentricity (.5 or more) it is easy to see that it is moving faster when it is closer to the sun. A much larger distance along the orbit is covered in one-tenth period than when the planet is far from the sun.

Below the picture are printed the semi-major axis (A) and eccentricity (E) you input and the planet's orbital period (P) in years.

The Third Law states that for any of the sun's planets

$P^2 = A^3$

if the planet's orbital period P is in years and its semi-major axis A is in A.U. Note that for earth, P=1 and A=1, so this law certainly works for OUR planet, anyway. Relations between squares and cubes are not particularly easy to picture, though (quick - what's A for a planet with a period of two years?). This program allows you to construct your own solar system (how's that for an ego trip?) to see the Third Law in action.

The program first asks you to input the number of planets you want to see, from 2 to 9. You don't have to use the numbers from the real planets unless you want to, but I limited this to nine planets because the program has to do so much calculation with more of them that it runs rather sluggishly. Type the number of planets, then hit RETURN. You are asked to input the semi-major axes (in A.U.) of your planets, one at a time. Type in each one, then hit RETURN. The computer draws the planetary system you've input, with the planets shown as dots and the sun as a somewhat bigger dot in the center of the screen. All the planets start out lined up to the right of the sun, like runners set to go around a track. The probability of this happening in real life is close to zero, but I did it so you'd have an easier task comparing how far each planet has moved compared to the others. As the Third Law suggests, the more distant planets take longer to go around the sun. This simulation allows you to see how much longer.

Below the picture is a "clock" that tells you how many years have passed. Time moves forward in steps equal to one-tenth of the orbital period of the innermost planet, and the simulation continues until the innermost planet has completed ten orbits.

# M A X W E L L - B O L T Z M A N N  D I S T R I B U T I O N

The Maxwell-Boltzmann Distribution gives the relative probability of a molecule of known mass having a speed near a particular value when it's in a material having some known temperature. It makes some intuitive sense. It turns out that it is very unlikely for a molecule to have a speed very much less or very much more than the average. This is similar to the statement that it is improbable that an adult be very much shorter or taller than the average, so it does seem to make sense. Mathematically, however, the statement looks quite formidable:

$$dP/dv = 4\pi \ (m \ / \ 2\pi kT)^{3/2} \ \ v^2 \exp(-mv^2 \ / \ 2kT)$$

dP/dv is related to the probability of finding a molecule with a speed near v. On the righthand side of the equation, m is the molecule's mass in kilograms, v is its speed in meters per second, T is the temperature in kelvins of the material the molecule is in, and k is Boltzmann's Constant.

This is the kind of equation that makes one think of changing one's major, but a graph of dP/dv versus v doesn't look complicated at all and lets you get a much more physical idea of what the Maxwell-Boltzmann Distribution is all about. This program does graphs of dP/dv versus v, with v ranging from 0 to whatever you wish.

The first thing you get asked for is the molecular mass in atomic mass units (amu). Your choice. It might be useful to remember that the average mass of an air molecule is about 29 amu.

You next are asked for the temperature in kelvins. Remember here that room temperature is about 300. Type the temperature, then hit RETURN. You are now asked to input the maximum v you wish to see on the graph. This is just how far you want the horizontal axis to extend, since it always starts at 0. Near room temperature, air molecules move at a few hundred meters per second on the average. This should give you a rough idea of what to choose for maximum v. Type it, then hit RETURN.

The computer now draws the graph of dP/dv for v ranging from 0 to your input maximum v. The probability of a molecule having a particular speed increases smoothly from zero for speed zero to a maximum at a speed called the most probable speed. It then decreases to smaller and smaller values at higher and higher speeds. If your maximum v is less than the most probable speed, you won't see the decline in dP/dv that takes place at high speeds.

Below the graph are printed six quantities. Three are your inputs: the temperature, molecular mass, and maximum v. The other three include the biggest value of dP/dv displayed, called "MAX DP/DV." This is usually very small, since not many molecules move at speeds very close to even the most probable speed. The last two numbers are the average and most probable speeds for molecules of the input mass at the input temperature.

It may seem mysterious that the average is not the same as the most probable - it's always bigger. The physical reason for this is that although high speeds are very improbable there are always a few molecules around moving much faster than the most probable speed. There are a few moving much slower, too, but the slowest they can go is just bigger than zero while there's no limit on how fast they can go short of the speed of light. This pushes the average up.

Properties like the most probable speed and average speeds depend on both the molecular mass and the temperature. I suggest that you treat this program somewhat like an experimental setup. Keep temperature constant for several runs while changing molecular mass (hydrogen to oxygen to carbon dioxide, say), then keep molecular mass constant while trying various temperatures.

If you run again by hitting A, you can keep the previous value of any input by hitting RETURN instead of retyping it.

One interesting application of this program involves looking up the temperatures, escape speeds, and atmospheric compositions of various objects in the solar system. Most astronomy books as well as almanacs and encyclopedias have this information. In light of what the Maxwell-Boltzmann Distribution can tell you about probable speeds of various molecules at different temperatures, you might see why earth has no significant hydrogen or helium in its atmosphere, the moon has no atmosphere at all, and Mars has only a thin one composed mainly of carbon dioxide.

# P - V  G R A P H S

This program draws graphs of thermodynamic changes of ideal gases in the P-V plane and calculates physical properties of the samples before and after the change you choose.

The program first asks you to input the initial pressure of your gas sample in pascals. 1 pascal is defined as 1 newton per square meter and is a very small pressure. The pressure of earth's atmosphere under standard conditions (called 1 atmosphere) is 1.013 E5 pascals. I mention this because one of my own most common blunders is to think of a standard atmosphere as 1 pascal--which leads to embarrassing results. Type the initial pressure, then hit RETURN.

The program now asks you to input the initial volume of the sample in cubic meters. Type this, then hit RETURN. Next, you are asked for the number of moles of gas. Type this, then hit RETURN. The next question is the nature of the ideal gas in the sample: monatomic or diatomic. Hit M for monatomic or D for diatomic. No RETURN is needed here.

Now that you've told the computer the pressure, volume, number of moles and type of gas in your sample, it asks you what kind of change you want the sample to undergo. Your choices are

1 ADIABATIC
2 ISOTHERMAL
3 ISOBARIC
4 ISOCHORIC

Hit 1, 2, 3, or 4 to choose. No RETURN is needed. The screen now turns white and the computer asks for a final input quantity. Depending on what kind of change you asked to see, this is either the final volume or the final pressure of your sample after the change. Type whichever is asked for, then hit RETURN.

The computer now draws the graph of the change you described, starting at the initial values of P and V and finishing at the final values. The graph shows you how (in terms of P and V) the sample changes its properties from its initial state to its final state during the particular process you input.

Under the graph are printed the initial and final values of P, V, and T. Below that is an invitation:

HIT D (DATA), G (GRAPH), OR Q (QUIT)

If you hit Q, the program gives you the usual instructions on what to do next. If you hit A to run again, you can keep the previous value of any input by hitting RETURN instead of retyping it. D zaps the graph and displays instead seven quantities, each preceded by the word DELTA (for "change in"). These describe the changes that have taken place from the initial to

the final state. They are, in order, the changes in P, V, T, and internal energy (U), the amount of work done ON the sample (W), the amount of heat ADDED TO the sample (Q), and the change in entropy (S). You can get the graph back by hitting G.

As usual, this program should be used with patience. Choose a particular initial P and V and a number of moles and stick with it while you try different kinds of changes. This is much like what you might do in lab if you had a particular sample of gas to experiment with. Running random sets of inputs won't teach you anything--you won't be able to see the patterns involved. You should probably start with standard conditions: P=1.013 E5, V=.0224, and 1 mole of diatomic gas (air is a good approximation to an ideal diatomic gas). Repeated runs of the program will give you some intuition for what is "big" and "small" as well. For example, is 100 joules a lot of heat to add to a mole of gas, or not much? Is an entropy change of 100 joules per kelvin a lot for one mole of gas, or very little? Having a good grasp on this can be very helpful. It will not only let you know if your answers to homework problems make sense, but will also help you later in your professional career when you are doing research.

# T H E R M O D Y N A M I C  C H A N G E S

This program produces timed quizzes on thermodynamic changes. It is intended to help you test your understanding of thermodynamics in a pressure (no pun intended) situation.

Your first choice is the level of difficulty. This controls how much time you get for each problem. The level ranges from 0 (about five minutes per problem) to 9 (about thirty seconds per problem). Even five minutes is not all that long for one of these, and thirty seconds requires the kind of superhuman speed found only in extraterrestrial beings and physics professors.

A quiz consists of a total of ten multiple choice problems. Your job is to solve each problem before its time limit expires and type the letter of the correct answer (A, B, C, D, or E). DON'T HIT RETURN, just the letter of the correct answer. You get credit only for correct answers. If you don't choose an answer before the allotted time expires, the computer goes on to the next problem.

Problems involve five different kinds of thermodynamic changes: adiabatic, isothermal, isobaric, isochoric, and free expansions. The type of change appears as a heading over each problem as the computer prints it. Seven different kinds of questions can be asked. You may be asked to find changes in pressure, volume, temperature, internal energy, or entropy, the amount of heat added to a system, or the amount of work done on a system. The system undergoing the change may be a monatomic or diatomic ideal gas. This already implies a vast array of possible problems, but to add to the fun the computer chooses the numbers given in each problem randomly. This gives you several billion possible problems, so a repeat performance is rather unlikely.

After you hit your guess for the answer (A, B, C, D, or E, and remember DON'T HIT RETURN) the computer displays the correct answer to the right of yours for a few seconds. It then automatically goes on to the next question. DON'T HIT RETURN. At the end of each quiz the computer displays the list of letters you hit and the right ones, along with your score.

# T H E R M O D Y N A M I C  C Y C L E S

This program allows you to design theoretical models of heat engines. These are devices that convert heat into work or, sometimes, work into heat. Applications of these models include machines ranging from diesel engines to refrigerators. A particularly important theoretical model of a heat engine is the Carnot engine.

The program assumes that the heat engine you are designing is a cyclic device. That is, starting from an initial state (given values of pressure and volume), thermodynamic changes take place until the state returns to the initial one - same pressure, same volume. It also assumes that there are exactly four steps that lead from the initial state back to the initial state. A final assumption is that the engine works by making changes in the state of an ideal gas. These assumptions aren't particularly restrictive, it turns out, and lots of real heat engines can be modelled by this program.

The program first asks you for the number of moles of gas. Type this, then hit RETURN. Next question is whether the gas is monatomic (M) or diatomic (D). Hit M or D (no return needed). The computer now asks you to choose the nature of the thermodynamic change undergone by the gas in each of the four steps in the cycle. Possible choices are

1 ADIABATIC
2 ISOTHERMAL
3 ISOBARIC
4 ISOCHORIC

Hit the number corresponding to the change you want to see (no RETURN necessary) for each of the four steps.

You are next asked to input the initial values of the pressure and volume. After typing each of these, hit RETURN.

The next steps involve typing the values of either pressure or volume (the computer will ask for the one it wants) after each of the first three steps. After typing each of these, hit RETURN.

You now may have run into a problem. At the end of the last step the computer expects pressure and volume to be the same as they were initially, since the assumption is here that the engine is cyclic. Unless you've worked out proper values of pressure and volume for your changes, the last change may not bring the engine back to its initial state. For example, suppose the initial volume is 1 cubic meter, you specified the volume after the third step to be 2 cubic meters, and you want the last step to be isochoric (volume constant). If the volume at the end of the third step is 2 cubic meters, and it stays constant during the fourth step, it will NOT end up being 1 cubic meter. Use of the program P-V GRAPHS may help you with this problem.

All this means that this program is going to take more work on your part than usual. This is actually not surprising, since designing a four-step heat engine requires a lot more information than a single thermodynamic change. More input quantities means more creativity is needed in working out good combinations of them, but also that the results are closer to practical application.

What you get from the computer in return for your efforts is a graph of your four step process in the P-V plane. Under this is printed the efficiency of the cycle, which is the net work done by the engine in one full cycle divided by the heat absorbed by the engine during the cycle. If we think of a heat engine as a machine that turns heat into work, higher efficiency means that more of each joule of heat absorbed (from burning fuel, for example) is turned into useful work. Efficiency can never be 100% (see the Second Law of Thermodynamics) even if we leave out depressing things like friction (as is done here). Some heat engines have negative efficiencies, sometimes on purpose. These are machines that convert work into heat. An interesting way to use this program is to see how efficient a heat engine you can design.

At the bottom of the screen is an invitation to hit D to see the data, G to see the graph, or Q to quit. If you hit D, the graph vanishes and you see a summary of the state of the system (P and V) after each step in the cycle, the work done on the system in each step, the heat absorbed by the system in each step, and the total work done on the system and total heat absorbed by the system for the entire cycle. Hit G to get the graph back again, or Q to quit. No RETURN is needed.

A note on signs: I consider work in a cycle to be positive if work is done ON the system (energy is added to it). Work is negative if work is done BY the system (energy is lost from it). Heat is positive if heat is absorbed by the system (energy is added to it) and negative if heat is lost from it. Although a lot of people use the same sign conventions as I do, some do not, and you need to be aware of that. It's not THAT confusing, but you've got to be ready for it.

This program is of a fairly sophisticated nature, and you should have a pretty thorough understanding of one-step thermodynamic changes before you try working with it.

# M O L E C U L A R  M O T I O N

This program allows you to perform simulated experiments on a gas of non-interacting molecules confined to a square region. One purpose of the program is to allow you to illustrate for yourself an important assumption made in thermodynamics. This assumption is that despite the random motion of individual molecules the net force on the walls containing a large number of molecules is nearly constant.

You get to choose a number of molecules to experiment on (1-40), a side length for their container (20-100), and a Kelvin temperature for them (0-1000). After typing each input, hit RETURN. The container for these molecules is a square. A real gas needs a three-dimensional container. This one is only two-dimensional, but the results will still make sense. Just don't apply them to 3-D gases without a little thought.

Once you've input the number of molecules, the size of the box, and the temperature, the computer displays several things. One is a scale model of the molecules in the box. The molecules are started by the computer at random places in the box, moving in random directions at speeds that depend on the temperature.

The computer moves each molecule once in each time step. The distance it moves depends on the temperature. At high temperatures, molecules move a long way during each time step; at low temperatures, the distance covered is comparatively small. Time steps represent very short times as measured in seconds. The air molecules you're breathing are moving at hundreds of meters per second, for comparison with speeds you see on the screen. This means that a measurement of a property like pressure would take many time steps - thousands, at least.

The simulation assumes that the molecules are non-interacting. That is, they move as if they are unaffected by the other molecules - in straight lines, at constant speed, until they hit a wall. You can see this best using a small number of molecules in a big box at low temperature. This assumption is OK so long as the gas being modelled is "dilute." This means there are fairly few molecules in a fairly big container, so the probability of two getting close enough to affect each other is low. Every time a molecule hits a wall, the computer will make a "click," so that you need only to listen to be convinced the collisions are really random.

The molecules' motion doesn't look smooth and isn't intended to. The computer calculates the molecules' positions at short but not infinitesimal time steps. The picture shows this directly.

The computer also prints a running total of collisions of molecules with walls that have taken place. Each tiny hit of a molecule against a wall contributes a tiny amount to the pressure, transferring some momentum to the container wall. In addition, the computer shows how many time steps have elapsed, and the ratio of total collisions to total time steps passed. This last quantity is a crucial one. Since a measurement of pressure takes

a lot of the tiny time steps, what a pressure gauge measures is closely related to the average number of collisions per time step, which is just the ratio referred to above.

The computer also displays a graph of this ratio versus time. You can see fairly easily that during the first few time steps the average number of collisions per time step is pretty random. Over many time steps, however, the average number of collisions per time step gradually settles down to a nearly constant value. This is a good indication that random collisions of molecules with container walls can indeed lead to the pressure being constant, or very close to it.

If you hit A to run again, you can keep your previous value of any input by hitting RETURN instead of retyping it.

There are other uses for this simulation as well. Treat it like an experiment. Keep two of the input quantities constant and vary the third over a number of runs until you understand the results. What happens to the collision rate if the number of molecules is decreased? If the temperature is increased? If the size of the container is increased? Try graphing the ratio of total collisions to total time steps against number of particles, temperature, and side length, in each case keeping the other two constant. Do the results make sense? You might find the program LEAST SQUARES FIT in the Mathematical Methods section useful here. Your text or lecture notes may have the development of the Ideal Gas Law, PV=nRT, from kinetic theory. If so, you can use this simulation to help you develop an analogous formula for two-dimensional ideal gases, which will give you some useful insights.

# E L E C T R I C  F I E L D

This program displays some of the field lines of a set of up to 10 charges. You get to input the number of charges, the amount of charge on each one, and the x- and y-coordinates of each one. Hit RETURN after each input. The one major restriction is that the absolute values of all x- and y-coordinates must be less than 1.

As you input charges and positions, the computer places a + or - at the appropriate point on the screen. Once all charges are in place, the program traces a set of field lines, indicating their positions by dots. Once this operation is complete, you see the invitation

D FOR DATA, F FOR FIELD, OR Q TO QUIT:

If you hit Q, you receive the usual instructions on what to do next. D zaps the picture and shows you a chart of the charges and positions that you input. F shows the charges and field lines again. You should bear in mind that the field lines drawn do not show the intensity of the electric field--only its direction.

The program is particularly useful in showing the fields of sets of charges such as dipoles and quadrapoles, but it can also illustrate much more complicated situations.

Some patterns of field lines, particularly for large numbers of charges, take several minutes to draw. If you have seen enough at any time during this process, hit any key to end the field line construction.

Not all of these are esthetically pleasing. If a set of field lines is not clear to you, try putting in the same charges, separating them by the same distances, but place them in a different part of the screen.

If you hit A to run again, you can keep the previous value of any input by hitting RETURN instead of retyping it.

# C A T H O D E  R A Y  T U B E

This program demonstrates the operation of a simple kind of cathode ray tube. Even the simplest of these is not all that simple, as you'll see, but those used in modern oscilloscopes and in color TV sets are a lot more sophisticated. That should be expected, since a lot of smart engineers have spent a long time improving the first primitive models.

A cathode ray tube is basically a device for producing a beam of electrons that are thrown at a screen coated with a substance that emits a flash of light when an electron hits it. The beam of electrons is produced by a device called an "accelerator" and is aimed by a device called a "deflector."

The accelerator consists of two pieces, called a "cathode" and an "anode." The cathode is a piece of metal that is heated electrically. The heating drives electrons off the surface of the metal. The anode, nearby, is kept at a high potential compared to the cathode, so the electrons driven off the cathode feel a strong pull towards the anode. A typical potential difference between a cathode and an anode is about 1000 volts, so the electrons are really trucking along by the time they get close to the anode--like ten million meters per second.

The anode has a small hole in its center, so that some of the electrons pulled off the cathode fly through the hole and continue onward, a relatively thin beam. If that were all there was between the accelerator and the screen, the cathode ray tube (or CRT) would always show a small bright dot in its center, where the beam passing through the hole in the anode hit it. That wouldn't be very useful, so a deflector is included, too. Actually, there are usually at least two, one to deflect the beam horizontally and the other one to deflect it vertically, but let's stay simple for now.

The deflector at its simplest is a parallel plate capacitor whose plates have a potential difference between them that is controlled from outside the CRT. The plates are arranged so that the beam of electrons passes between them on its way to the screen. If there is a potential difference between the plates, the resulting electric field will deflect the beam of electrons as they pass between the plates. They don't stay between the plates very long (remember how fast they're moving) so the electric field doesn't have too much time to act on them, and they usually get deflected through a fairly small angle as a result.

In the demonstration, the accelerator is at the left of your screen. The cathode is represented by an L-shaped figure, the anode by a line with a hole in it to represent the hole in a real anode that lets electrons through. When the program first starts up, that's all you see in the picture. After you've typed some inputs (described below) the rest of the apparatus will be drawn. One part of this is the deflector. This will be a little to the right of the cathode and anode, and will be represented by a pair of parallel lines, one above the other, to suggest two parallel plates seen edge-on. On the extreme right of the screen will be a long vertical line representing the screen of the CRT, also seen edge-on.

You get to control five input quantities. Three of these have to do with the CRT itself, the other two with the external potential source that controls the potential difference between the deflector plates. The three characteristics of the CRT that are under your control are the accelerating potential (the potential difference between the cathode and anode), the distance between the deflector plates, and the length of the deflector plates. The bigger the accelerating potential, the faster the electrons move. You can choose this to be any value between 100 and 10,000 volts. Hit RETURN after typing it.

The distance between the deflector plates affects the strength of the electric field between them. The further they are separated, the weaker the field and the less the electrons will be deflected. You can choose this to be 1-10 millimeters. Hit RETURN after you type it.

The length of the plates affects how long a time the electrons are in the deflecting electric field. The longer the plates are, the more time it will take the electrons to pass between the plates and the more deflection the electrons will feel as a result of the deflecting electric field. The length of the plates can be anything between 5 and 50 millimeters. Type your choice, then hit RETURN. After this input, the deflector plates and screen are drawn.

The last two inputs are the amplitude and frequency of the potential difference between the deflector plates. The program assumes that this voltage varies sinusoidally with time, deflecting the electrons upward, then downward, then upward again. The maximum deflection depends on the amplitude of the deflecting potential. This can be 0-1000 volts. Type this, then hit RETURN. How fast the deflection switches from up to down and back to up again depends on the frequency of the deflecting potential, which can be 50-500 hertz. Type this, then hit RETURN.

The program shows you a glimpse of the inside of the CRT every .0001 seconds. The picture is slowed down from real speed by a factor of about a million so you can see the dot representing an electron zoom from the cathode to the anode, through the anode, between the deflector plates, and on to the screen at the right of the picture.

I suggest that when you first start running this you choose the frequency to be 50. This will allow you to see more easily the effect on the path of an electron caused by changes in accelerating potential, deflecting potential, and the separation and length of the plates. As usual, you should keep all but one of the inputs constant while running the program for several choices of the remaining quantity, until you understand how it influences the behavior of the electron.

You can end a particular experiment at any time by hitting RETURN. If you run again by hitting A, you can keep your previous choice for any input by hitting RETURN instead of retyping it.

# U N I F O R M  B  F I E L D

This program allows you to experiment with the behavior of an electron in a uniform magnetic field. You will have available four input quantities: the strength of the magnetic field, the kinetic energy of the electron, the angle between the electron's velocity and the direction of the magnetic field, and the angle to the magnetic field direction from which you wish to view the electron's motion.

The magnetic field strength is measured in teslas, and your choices are .1-10. A tesla is a pretty big unit; earth's magnetic field is about 5 E-5 teslas. The electron's energy is measured in units called KeV (for kilo-electron volts). 1 KeV = 1.60 E-16 joules. This doesn't seem like much, but it's quite a bit to a single electron. Your choices here are 1-100. This energy basically determines the electron's speed. The angle between the direction the electron is moving and the direction of the magnetic field is 0-180 degrees. At 0 degrees the electron is moving in the same direction the field points. At 180 degrees, the direction of the electron's motion is opposite to the direction of the field. A choice of 90 degrees means the electron is moving perpendicular to the field direction. The final input is the "tilt angle," which defines the direction from which you view the motion of the electron. This can be between 0 degrees (looking in the same direction as the magnetic field) and 180 degrees (looking opposite the direction of the field).

After typing each input, hit RETURN.

The path of an electron in a uniform magnetic field is a helix, the shape of a spring or a DNA molecule. By trying different inputs you can see what effect they have on the electron's trajectory. What happens if you leave the field at 1 tesla and try various energies? What happens if you leave the energy at 10 KeV and change the field strength? I suggest that you run the program the first few times with both angles equal to zero for simplicity. Once you understand to some extent how the field strength and energy influence the behavior of the electron you can experiment with different angles. To help you see the path in three dimensions, the position of the electron is plotted slightly differently when it is on "your side" of the TV screen from when it is on the "far side" of the screen.

The formula for the force F produced on a particle of charge q moving at velocity v in a magnetic field B is given by the Lorentz force equation

$$F = q\ v \times B$$

The cross product of the two vectors v and B that appears in this formula makes the behavior of F under various conditions very strange. The angle between v and B is crucial in determining the electron's motion. You should vary this while keeping the field strength and energy constant. By changing the tilt angle you can change the direction from which you are viewing the electron's path. The computer lets you change this by 5 degrees automatically if you hit T (no RETURN needed) after the simulation is completed.

Incidentally, the program sometimes runs for a few seconds after the electron has drifted out of view. Wait for instructions before you hit any keys (you'll also hear a beep).

The electron's speed is slowed by a factor of about ten billion and the picture expanded by a factor of about a million so you can see what's going on.

The program prints the three input quantities below its picture of the moving electron. It also prints the radius of the electron's helical path (in meters), the component of the electron's velocity along the direction of the magnetic field (in meters per second), and the cyclotron frequency. This last is the angular frequency (in radians per second) of the part of the electron's motion that involves circling around the direction of the magnetic field. Both this and the speed in the direction of the magnetic field are usually large numbers.

If you hit Q to quit, then A to run again, you can keep the previous value of any input by hitting RETURN rather than retyping it.

# L R C  C I R C U I T S

This program allows you to examine some of the properties of LRC circuits by displaying a graph of voltage versus time for any of the circuit elements. The voltage source in the circuit is assumed to be set at 0 until time t=0, at which point it jumps suddenly to 100 volts. This may be thought of as modelling a switch being thrown or as a positive jump produced by a square wave generator. The circuit elements are assumed to be arranged in series with the voltage source.

You are first asked whether a capacitor is present. You should respond by hitting Y (for yes) or N (for no). No RETURN is needed. If you type Y, you will be asked to input the capacitance, in farads. A farad is an enormous unit. Most lab capacitors are around a microfarad ($10^{-6}$ F or 1μF). After typing the capacitance, hit RETURN.

You are now asked if there is a resistor in the circuit. As before, hit Y or N (no RETURN needed). If you hit Y, you are asked for the resistance in ohms. A moderate value for this is 100. Type the resistance, then hit RETURN.

Next, you are asked if the circuit includes an inductor. Respond with Y or N, no RETURN needed. If you respond with Y, you are asked to input the inductance in henrys (if you think this is a strange name for a unit, ask a particle physicist about barns and jiffies, or a radio astronomer about janskys. It takes all kinds.) Inductances are usually around .1 henrys. After typing the inductance, hit RETURN.

One warning: you must have at least two of the three possible circuit elements present. The program won't do a lone resistor, for example.

After entering the properties of the circuit elements you are asked which element you wish to measure the voltage across. Respond with L, R, or C (no RETURN is needed). Finally, you are asked for the total time you wish to see displayed on the horizontal axis. If you choose something huge (1 E10 seconds, for example) your graph won't show you anything interesting. You should experiment with different times to see interesting features. Important times for electric circuits tend to be short, however, as measured in seconds, so .01 as a first guess might not be bad. This does depend strongly on your inputs for L, R, and C, however.

If you hit A to run again, you can keep the previous value of any input by hitting RETURN instead of retyping it.

This program is intended to help you gain some insight into the workings of LRC circuits. A key analogy to make is with damped harmonic oscillators. This is pointed out by most instructors, so I won't belabor it, but you might run DAMPED HARMONIC DEMO in the Mechanics section to see the graphical similarities. Mechanical systems like masses bouncing on springs are subject to direct sensory examination (i.e. you can see them, hear them, touch them, and, if sufficiently kinky, even taste them) and so you can have some

confidence that you understand how they work. The mathematical descriptions are the hard part. When it comes to LRC circuits the situation is reversed. Looking at one, or touching it, tells you nothing useful about its electronic behavior. You are reduced to measuring its properties with oscilloscopes and trying to understand it by the intellectual process of grasping the mathematics. It is in the EQUATIONS describing LRC circuits that we find analogies with damped harmonic oscillators. The analogy (a literature student would call it a metaphor) is a very sophisticated one, in my opinion. Don't let the similarity between the equations

$$L\ d^2I/dt^2 + R\ dI/dt + (1/C)\ I = 0$$

$$m\ d^2x/dt^2 + b\ dx/dt + k\ x = 0$$

and their solutions disguise the fact that it is rather amazing that a mass bouncing on a spring has something in common with an electric circuit. There are other and even more surprising phenomena described by the same form of equation. You'll run into them in upper level physics courses later on.

For convenience, the graph of V versus t displayed has markings on both axes. The t axis is divided into tenths. The V axis has marks at +100 volts and -100 volts. Remember that I'm assuming the circuit is "dead" (no voltage differences anywhere) until t=0, when a 100 volt potential is suddenly turned on in series with your circuit elements. Using the damped harmonic oscillator analogy, this is rather like starting out with a mass hanging motionless from a spring, then suddenly increasing the gravity in the room to a higher value. The mass plunges downward, bounces for a while, then settles down at a lower level than before. Like the bouncing spring, you find the voltage across an element of an LRC circuit overshoots its equilibrium value, returns, and gradually settles down to a constant value.

Something you can exhibit here that I left out of DAMPED HARMONIC DEMO is behavior in critically damped and overdamped situation where no oscillation occurs. A big enough resistance (analogous to the damping coefficient in a damped harmonic oscillator) will always lead to overdamped behavior. It's not very exciting, but you should see it.

I expect that this program will be of some use to you when you deal with LRC circuits, oscilloscopes, and function generators in lab. The program is much simpler, and should provide a kind of bridge between theoretical work in class and experimental work in lab.

# E L E C T R I C A L  G E N E R A T O R

This program demonstrates a very simple model of an electrical generator. By Faraday's Law, the changing of magnetic flux through a loop of wire will induce a voltage in the loop which will produce an electric current. This program allows you to input a strength for the magnetic field (assumed to point directly out of the screen), a length of side of the loop (assumed to be square), and a frequency of rotation for the loop. As the loop rotates around an axis running horizontally across the screen, the magnetic flux through it will change, inducing a voltage called the "electromotive force" or EMF.

The magnetic field B can have any value from 1 to 5 teslas. Hit RETURN after you enter it. Since the EMF (measured in volts) is proportional to B, you should try small B and large B (keeping the other inputs constant) to see this effect. The EMF is also proportional to the area of the loop, since this affects the magnetic flux through the loop. I throw you a curve here by having you input the side of the loop rather than the area (values can range from .02 to .1 meters). I did this because it's fairly easy to estimate lengths, but harder to estimate areas. In getting an idea of how loop size affects the output, it's easier to examine the lengths of the sides of the loop. After typing side length, hit RETURN.

Finally, type the frequency of rotation of the loop in hertz. This can be any number 1-5. Hit RETURN after typing it. Since the EMF depends not on the flux itself but on how fast the flux changes, the bigger the frequency of rotation the larger the voltages that will be obtained. The flux is largest in absolute value when the loop lies in the plane of the screen. It looks like a square at this time. The flux through the loop is zero when the loop is seen edge-on. It looks like a line in this situation.

Once you've input the magnetic field, side length of the loop, and frequency of the loop's rotation the simulation starts. On the left side of the screen is the rotating loop. Its size depends on what you input for side length. It will appear to rotate (use your imagination a bit; I did my inartistic best) at a speed that depends on the frequency you input.

The rest of the output consists of two graphs. The lower one is a graph of magnetic flux $\Phi$ through the loop as a function of time. You should note that when the loop is edge-on to you, and hence to the field, the flux through it is zero. When the loop is face-on to you, so it appears as a square on the screen, the flux is at maximum, either positively or negatively depending on whether the loop is facing toward you or away from you. The maximum possible flux depends on the area of the loop and the strength of the magnetic field, but not on the rotation frequency.

The top graph shows voltage (EMF) versus time. The voltage depends on the rate of change of the flux, so it is zero when the flux is at maximum or minimum. It is biggest, either positively or negatively, when the flux is increasing or decreasing most rapidly. By watching the sketch of the loop, it is easy to see when the flux is changing rapidly--the size of the

picture of the loop is getting bigger or smaller rapidly. When the loop is face-on, its size is changing slowly, so the voltage is near zero.

All graphs are to the same scale. The top of the vertical axis in the lower graph is at .05 webers (1 weber = 1 tesla-meter squared. The top of the vertical axis of the upper graph is at 1.57 volts. The time axis is 1 second long.

One variable I didn't include here is the number of turns of wire in the loop, which I took to be 1. The reason is that if I let you vary this, too, the scaling on the graphs becomes useless. If maximum possible values were to fit on the graphs, most smaller ones would be too small to rise above the time axis. This is even some problem as things stand. With small values of side length, only the biggest magnetic fields and frequencies show visible results on the graphs.

When you run the program again by hitting A, you can keep your previous choice for any of the inputs by hitting RETURN without retyping it.

Wave Phenomena

# W A V E S

This program allows you to input the amplitude, frequency, and phase of a wave. The computer then graphs the wave's height, Y, versus time, t.

All of the waves here are of the mathematical form

$$Y(t) = A \cos(2\pi ft + \emptyset)$$

where A is the amplitude, f the frequency, and $\emptyset$ the phase. When I first saw this equation as a freshman, all the individual pieces made sense, but the combination was too complicated for me to get a good grasp on. I can recall instructors blithely talking about two waves with phases different by 90 degrees, or two waves 180 degrees out of phase, but I couldn't translate that smoothly into a physical picture. Hopefully, this program will allow you to mess around with amplitudes, frequencies, and phases until they DO make sense. They aren't really very complicated (really!) so long as you've got a picture of what's going on - and that's what the program draws for you.

The first thing you get to choose is amplitude. Your choice has to be in the range 0-100. I deliberately avoided putting units on this, because there are too many possibilities. 100 is a big amplitude here; the waves about reach the top and bottom of the graph. Amplitude 0 is a "wave" of constant height, and that height is zero. The graph of this lies entirely on the y-axis, and you don't see anything interesting. After typing the amplitude, hit RETURN.

Next, input the frequency. This is 0-10 hertz. The time axis extends from 0 seconds to 1 second, so a wave with frequency 1 will make one complete cycle. A wave with frequency 10 will make ten complete cycles. If you type 0 for the frequency, the result is a "wave" whose height is always the same. Notice that in this case the coefficient of time in the equation is zero, so it doesn't matter what t is, the value of y is always the same. After typing the frequency, hit RETURN.

Finally, you need to input the phase, in degrees. In the formula, phase should really be in radians, but the program automatically converts what you input from degrees to radians. I did this because most people, me included, seem to think better in degrees than in radians, although radians are easier for theoretical work. Hit RETURN after typing the phase.

Phase controls where the wave starts at t=0. Amplitude tells you how high it gets, frequency how close together the peaks are, and phase the place on the wave that is passing you at t=0. The best way to get a grasp on this is to keep amplitude and frequency constant and run the program repeatedly with different phases, say 0, 45, 90, 135, and 180 degrees, for starters. Phases of 90, 180, and 270 degrees are the most important physically, because they lead to the simplest results. Try it and see.

The best way to use this program is to start by picking a frequency, say

1, and a phase, say 0, and keeping them the same while trying several different amplitudes. Presently this will begin to make sense. Then choose an amplitude, say 100, and a phase, say 0, and try several different frequencies until you understand what effect this has (count the total number of waves plotted). Finally, pick a constant amplitude (100, for example) and frequency (1 is a good choice) and try various phases. With a few runs, waves will hopefully be a lot clearer to you.

If you run the program again by hitting A, you can keep your previous choice for any input by hitting RETURN instead of retyping it.

# S U P E R P O S I T I O N

This program allows you to input the amplitudes, frequencies, and phases of up to twenty waves. The superposition (sum) of all the waves is then plotted on a graph. If you have any confusion about amplitude, frequency, or phase as applied to a single wave, you should definitely run the program WAVES a few times before tackling this one.

In addition to the number of waves and their amplitudes, frequencies, and phases, you can adjust the scale of the time axis, which is the horizontal one in the graph. (Hit RETURN after typing each input). The computer asks you for the total time to be displayed (in seconds). Choose whatever you wish. If you choose a huge time, however, your graph may be so squashed it makes no sense. Similarly, if you choose a very small total time your graph may be so spread out that all you see is a tiny section right after t=0. You'll have to experiment with this to make it give pleasing results. It is a LOT easier than doing the job on graph paper, point by tedious point, looking up all the cosines in tables. See how we suffered in the Old Days?

Once the graph is plotted, you are invited to hit D for the data, G for the graph, or Q to quit. If you hit D, the graph is replaced by a table of the amplitudes, frequencies, and phases of the waves that you input. G brings the graph back again. Q gives you the usual choice of running again or returning to the menu.

With twenty waves superposed, the program takes around four minutes to run. Smaller numbers of waves run faster, since the computer has to calculate fewer cosines.

If you run again by hitting A, you can keep your previous choice for any input by hitting RETURN instead of retyping it. This can be very helpful in some applications (see below).

All of the waves are assumed to be of the form

$$y(t) = A \cos(2\pi f t + \emptyset)$$

where A is the amplitude, f the frequency (in hertz), and $\emptyset$ the phase. The phase you enter should be in degrees. The computer converts this automatically to radians for use in the program.

The computer evaluates the sum, or superposition, of all your waves at intervals of 1/250 of your total time. It is no more capable of evaluating a function at ALL times from 0 to your total time than you or even I. This sometimes leads to rather jerky-looking results, and you should be ready for this. In particular, if you choose too large a total time your results may be a mess. If you can't see what's going on, run again with a smaller total time. Every point plotted is correct, but you may have to squint a bit to see the overall features. The vertical axis, incidentally, is automatically scaled so that the distance from the time axis to the top of the screen is equal to just slightly more than the sum of all your amplitudes. This guarantees that the results will always fit on the screen.

The program can be used several ways, and I'll suggest three here. One way is to illustrate the phenomenon of beats. I suggest that for this you choose two waves of equal amplitudes and phases. Both amplitudes equal to 1 and both phases equal to 0 is a simple possibility. Choose the frequency of one wave to be 100 hertz and that of the other to be just over this, say 101 or 102. Try a total time equal to 1 second. The output is then the total of the two waves. You can see that over a time of one second the maximum of the total starts out fairly large, decreases, then grows again. This also works fine for other amplitudes, frequencies, phases, and total times, but I wanted to give you at least one to start off with.

A second application is to examine the results of adding waves of equal frequencies but different amplitudes or phases or both. In fact, one of the simplest and most important demonstrations of superposition you can do is to take two waves of equal amplitude and frequency and see what happens if they are added together with different phases. If you choose both amplitudes to be 1, both frequencies to be 10 and the phase of one to be 0, you can get some interesting results by choosing the phase of the second wave to be 0, 45, 90, 135, and 180 degrees on successive runs. Take total time to be 1 second. If complete destructive interference takes place, the height of the superposed waves will be zero and the graph will lie right on top of the time axis.

A third application is included for those among you who have run into Fourier series. If you haven't already, you will in the future. This program provides a way of adding together terms in such a series to see what the results look like graphically. It is not at all hard to use it on Fourier cosine series, since the waves are assumed to be cosine waves. Input the Fourier coefficients for the amplitudes and the Fourier frequencies for the frequencies, and let all phases be 0. The amplitudes can be chosen to be negative numbers if necessary; the program can handle this fine. For sine series, choose the amplitudes and frequencies as for the cosine series, but take all the phases to be 270 degrees. This choice will convert the cosines to sines.

As an example, try the series

$$f(t) = \sum_{n=0}^{\infty} [1/(2n + 1)] \quad \sin[2\pi\ (2n + 1)t]$$

which represents a square wave with period 1, amplitude $\pi/4$. This is not at all obvious from the series, but the program will show it nicely. Tell the computer you wish to input 20 waves, which will give a good approximation to the function. The frequencies of the successive waves are the first twenty odd integers: 1, 3, 5, ... The corresponding amplitudes are the inverses of these: 1, .3333, .2, ... All of the phases should be 270 degrees, since this is a sine series. If you choose the total time to be 2 seconds (two periods) you see a good example of what a Fourier series does in representing a function. The graph even shows the Gibbs phenomenon, an overshoot of the function being represented that occurs near discontinuities. This looks like small "ripples."

# S N E L L ' S  L A W

This program allows you to simulate what happens when an incident ray of light hits the interface between two transparent media, like air and glass. You get to input the index of refraction of each medium and the incident angle of the ray as it hits the interface. This angle is measured from the normal to the interface at the point where the incident ray hits it. After typing each input, hit RETURN.

The computer draws a diagram of the situation your inputs describe. Medium 1 is on the left of your screen, medium 2 on the right. They are separated from each other by a black vertical line. A dotted line running horizontally across the picture represents the normal. The computer plots (in order) your incident ray, the reflected ray (bounced back into medium 1), and the refracted ray (if there is one) transmitted on into medium 2. The simulation then prints information below the picture, including your inputs as well as the angles of the reflected and refracted rays. The angle of the reflected ray is always equal to that of the incident ray. The angle of the refracted ray is calculated using Snell's Law, hence the name of the program.

A good way to use this program is to choose the index of refraction of each medium and to keep these constant while slowly varying the incident angle from one run of the program to the next, say by ten degrees each time. This will give you some feeling for refraction in various circumstances and also let you experiment with finding the angle beyond which total internal reflection occurs. This last phenomenon happens only if the refractive index of medium 1 is greater than that of medium 2, such as is the case for light passing from water to air or glass to air.

If you hit A to run again, you can keep your previous choice for any input by hitting RETURN instead of retyping it.

# T H I N  L E N S

This program simulates an experiment with a thin lens. You get to input the focal length of the lens (either positive or negative), the size of the object, and the distance of the object from the lens. The computer draws a representation of the lens, its optical axis, the object, and the image produced by the lens. The focal points of the lens are indicated by small hashmarks on the optical axis. The object is considered to be a nail (my artistic skill is very limited) with the flat head towards the top of the screen.

The object size, image size, focal length, object distance, and image distance are all drawn to the same scale. This can lead to problems if you attempt to experiment with a gigantic object very far from the lens (as one example). Try to keep your inputs within the same factor of ten of one another. If you don't, nothing awful happens, but the picture may not be of any use to you.

Once you've typed your inputs (hit RETURN after each one), the computer draws a sketch of the situation and prints some information below it. This information includes your three inputs along with image size and image distance. Image size is negative if the image is inverted. Image distance is negative if the image is on the same side of the lens as the object. Pay attention to these numbers so you can be sure which "nail" in the sketch is the object and which one the image. The object is always upright (flat head upward) and to the left of the lens, but sometimes this is true of the image as well.

Below the numerical data is printed

L TO MOVE LEFT, R RIGHT, OR Q TO QUIT.

If you hit Q, you get the usual instructions on what to do next. If you hit L, the object is moved to the left. Its distance from the lens increases by 10% and the sketch and numbers are redone. If you hit R, the object is moved to the right. Its distance from the lens is decreased by 10% and the sketch and numbers redone. This feature will allow you to get some intuition for the behavior of images as the distance between the lens and the object is adjusted.

If you run again by hitting A, you can keep your previous choice for any input by hitting RETURN instead of retyping it.

# R A D I O A C T I V I T Y

This program allows you to simulate an experiment involving a radioactive source and a shielding material. The experiment allows you to calculate a property of the atoms of the shielding material referred to as their "cross-section." A material whose atoms have a large cross-section is a good absorber of radioactive emissions. Lead is a good example.

If the number of particles emitted by a radioactive source in some period of time is called $N_0$, the number N which penetrate an absorber is less than $N_0$. The relation between N and $N_0$ is

$$N = N_0 \exp(- n s x)$$

x is the absorber's thickness in meters. n is the number of atoms per cubic meter of the absorber, called the "number density" of the absorber's atoms. s is the cross-section of the absorber's atoms in square meters, the quantity we wish to determine in this simulation. The relation between N and $N_0$ can be written differently by taking the natural log of both sides:

$$\log N = \log N_0 - n s x$$

and this is the form we will use here.

When the program is run, you are shown a sketch which includes a radioactive source (at left) and a detector (at right). You might think of the detector as a Geiger counter. Below the sketch are printed the density D and atomic weight A of the absorber. These can be used to find the number density:

$$n = 6.02 \text{ E26 } D/A$$

D and A are chosen randomly by the computer and don't correspond to real materials.

You are first asked to input the absorber thickness, which must be in the range .1 to 1000. The thickness is in microns; 1 micron is 1 E-6 meter. It is best to start with .1 microns for absorber thickness. Once you've typed this, hit RETURN. The absorber is now sketched in between the source and the detector.

N is the number of particles emitted from the radioactive source that reach the counter by penetrating the absorber. The computer counts these for a time of your choice, ranging from 1 to 100 seconds. Radioactive emissions are random, so that if the source is not very strong (not many emissions in an average minute) and the counting time is small you might get a very distorted value for N. If the source happens to emit a lot fewer particles than average or a lot more, it can throw off your calculation of s. This can be helped by using a longer counting time, so the randomness has time to average out. On the other hand, a strong source (lots of emissions in an average minute) may let you use a shorter counting time--

saving YOU time. The strength of the source is chosen randomly by the computer, so you have to make a choice for counting time that seems reasonable to you. 1 second is almost always too short; 100 seconds is usually longer than necessary. 10 seconds is a good first choice. If for absorber thickness .1 microns and counting time 10 seconds you get fewer than 10 counts, you should choose a longer counting time. After typing the counting time, hit RETURN.

You now see the particles (this is a simulation, remember!) shooting out of the source. Some of them penetrate the absorber and hit the counter. When this happens, you hear a click. The computer keeps track of passing time and the number of emissions (counts) detected.

Once the counting time is over, the computer asks you if you wish to try another absorber (hit A) or quit (hit Q). No RETURN is needed either way. If you hit A, the computer asks you for absorber thickness and counting time as before. You may wish to adjust your counting time after the first run, but successive measurements of N can only be compared if the counting time is the same for each one, so once you've found a good choice, stick to it.

Assuming you've found a choice of counting time that gives a value of N of at least 10 for absorber thickness .1 micron, you should proceed by running the experiment repeatedly. Counting time should be the same on each run, but you should make the absorber thicker on each run until the number of counts is close to zero. The computer lets you run as many as ten experiments.

After the tenth run, or if you hit Q at the end of an earlier run, the computer displays the data gathered by your experiments. This includes absorber thickness in microns, counting time in seconds, and the number of counts obtained for each thickness and counting time. The computer also displays the absorber's density and atomic weight and the "accepted value" of the cross-section. This last is provided as a check on your own calculation of s. A method for doing this calculation from the available data will now be described.

Since radioactive decay is a random process, algebraic solution of

$$\log N = \log N_0 - n\ s\ x$$

for s won't work. Even experiments with the same values of n, x, and counting time will not usually give the same values for N. For this reason, a method is used that helps to average out the randomness. This method uses the LEAST SQUARES FIT program.

From the data the computer provides, it is a simple matter to find the natural log of each value of N using a calculator. Each log N corresponds to a value of absorber thickness, x. We might think of x as the independent variable and log N as the dependent variable in this equation. The value of $\log N_0$ is a constant, even though it is unknown. The values of n and s are also constant for all the runs. This means that the equation

$$\log N = \log N_0 - n\ s\ x$$

is a linear relation. If log N is plotted on the vertical axis of a graph and x on the horizontal axis, the result is a straight line with intercept log $N_o$ and slope -ns.

Your procedure should be as follows. First, make a table of log N and x (remember that's NATURAL log N and that the thickness in microns needs to be multiplied by 1 E-6 to convert to meters). Next, use the values of density and atomic weight to calculate number density n. Finally, write down the "accepted value" of the cross-section s. Now hit L to run LEAST SQUARES FIT. Enter the values of x as x-coordinates and the corresponding values of log N as y-coordinates. The computer responds by plotting your data and giving you the least squares fit values of slope and intercept for the resulting straight line.

The slope of this line is -ns, a large negative number. Since n is known, the value of the slope can be used to find s. This value, obtained by your simulated experiment, should be compared to the computer's "accepted value" to make sure you've made no major blunders in all this. WARNING: The two values of s--yours and the "accepted value"--will NOT be the same, though they usually differ by less than 50%. Accepted values of cross-sections are obtained by doing large numbers of experiments with long counting times to average out random factors.

# S P E C I A L  R E L A T I V I T Y

This program demonstrates the behavior of time-measuring devices moving at high speed relative to one another. The program displays two "clocks." The lower one is at rest with respect to you, the upper one moving to the right at a speed determined by an input quantity called "beta."

Beta is the relative speed of the two clocks divided by the speed of light (v/c). If the relative speed is very small compared to the speed of light, beta is small (say .01). If the relative speed is close to the speed of light, beta is nearly (but never quite) 1. This program allows you to choose beta between 0 and .999.

The two systems shown represent simple clocks. Since I have only vague ideas about how real clocks work, I use ones whose only moving parts are photons. Each clock consists of a pair of mirrors (ideal, perfectly reflecting ones) with a photon bouncing back and forth between them. Each time the photon hits a mirror, you hear a tick.

The first postulate of special relativity is that the speed of light is the same in all inertial systems. For this reason, the program carefully makes the photons in both clocks move the same distance between ticks of the lower clock. The lower clock is at rest with respect to you, so the photon in that clock just bounces up and down vertically between the two mirrors with no horizontal motion. The upper clock is moving to the right, however, at a speed proportional to beta (your input). Hence the photon in this clock is moving both vertically and to the right. Since both photons must have the same speed from our viewpoint, they must move the same distance on the screen in the same period of time. The photon in the upper clock must cover some distance to the right in addition to some in the vertical direction, so it doesn't cover as much distance in the vertical direction in one second as the photon in the lower clock does.

Since the mirrors in both clocks are the same distance apart in the vertical direction, the photon in the lower clock hits a mirror (and makes a tick) somewhat more often than the one in the upper clock. Ticks of the upper clock are further apart than ticks of the lower one. The faster the upper clock moves to the right (bigger beta) the further the photon in the upper clock has to move to the right each second, and the less vertical motion it can accomplish. The faster it moves to the right, the longer it is between its ticks from our viewpoint.

From our viewpoint, then, the clock that is standing still with respect to us seems to tick more often than the clock that's moving with respect to us. So far as we can tell, the moving clock doesn't count as many seconds passing as our own, stationary one. Which clock is keeping the "right" time? I think I'll leave that one to your instructor to answer.

The program includes "digital readouts" of both clocks, counting their ticks for you. It also prints out the dilation factor. This is calculated from the standard relativistic formula relating passage of time in two

systems moving with respect to each other at speed indicated by a value of beta. The factor is

$$\sqrt{1 - \beta^2}$$

Incidentally, it's not just strange time-keeping devices like the photon-mirror clocks that experience this effect. All clocks, even biological ones, act this way. This might be a bit easier to swallow if you realize that the bottom clock (standing still with respect to you) really is a perfectly good clock. You could use its ticks to estimate your pulse rate, for example. If your pulse throbbed once for each tick of the bottom clock, someone moving at high speed with respect to you would see your clock running slower than his or hers--and your pulse rate, too.

This seems very weird, and it is. It's totally outside our everyday experience, and physicists in the early years of this century found relativity very hard to believe. The theory has been tested by experiment thousands of times, though, and it really works. The clocks in systems moving very fast with respect to observers really do seem to run slow by the factor predicted by the time dilation formula.

# H Y D R O G E N  S P E C T R U M

This program allows you to see the spectral lines produced by some of the transitions from high to low energy states in hydrogen atoms and also allows you to see the effects of Doppler shift on a spectral line's wavelength.

Atoms can emit photons when their energy changes from a higher to a lower value. Such an energy change is called a "transition." The various possible energies of hydrogen atoms (often called "energy levels") are numbered starting at 1 for the lowest (called the "ground state"), 2 for the next lowest, and so forth. The symbol for the energy level is n, and this is an input quantity. You input both the initial n (higher level) and final n (lower level). Follow each by hitting RETURN.

The computer plots a picture of a spectrum (not in color, alas) with the spectral line corresponding to the transition from higher n to lower n marked as a vertical line. Above and below the specrum (a horizontal band) are small hashmarks corresponding to particular wavelengths, so you can get an idea of the relative position of various spectral lines. The hashmarks start at 1000 Angstroms at the left end of the spectrum and are placed at 1000 Angstrom intervals to 9000 Angstroms near the right end of the spectrum. 1 Angstrom (= 1 E-10 meters) is a very commonly used unit in spectroscopy. Longer hashmarks denote the limits of the visible part of the spectrum. The lefthand longer hashmarks are at 3500 Angstroms (violet), the righthand ones at 7500 Angstroms (red). To the right of the visible part of the spectrum is the set of "colors" called infrared, which your eyes don't detect but your skin does - it's just heat. To the left of the visible spectrum are the ultraviolet wavelengths. They are also invisible, but can be detected in other ways, for example by producing sunburn in lightskinned people.

Below the spectrum are printed your inputs (initial and final n), the wavelength of the photon emitted (in Angstroms, symbol A), and the energy of the emitted photon. This energy is given in electron volts (eV), a common unit of energy used in atomic physics. 1 eV = 1.60 E-19 joules, but it may be more useful to know that photons of visible light have energies between about 1.65 eV (deep red) and 3.55 eV (very violet). Infrared photons have less energy, ultraviolet more. The energy of the emitted photon is equal to the difference between the energy of the hydrogen atom in its initial state and its energy in its final state.

Below the numerical output is printed

MORE LINES (M), DOPPLER SHIFT (D), OR QUIT (Q):

If you hit Q (no RETURN needed) you get the usual instructions on how to proceed. If you hit M (no RETURN needed), the program lets you input another initial n and final n while keeping the spectrum with the line already displayed. You can repeat this as many times as you wish, adding more and more lines to the spectrum. In this way you can see the set of spectral lines produced by hydrogen atoms falling to final energy state n=1 from initial

states n=2, 3, 4, ... This set of lines is called the Lyman series. The set produced by hydrogen atoms falling to the final energy state n=2 from initial states n=3, 4, 5, ... is called the Balmer series, and is of great importance in astronomy and astrophysics. After adding any new line or lines to the first one drawn, you lose the option of Doppler shifting your first line. The computer prints

MORE LINES (M) OR QUIT (Q):

under the numerical data on the last line you added to the spectrum.

If you hit D (no RETURN needed) after inputting the initial n and final n for your first line, you see printed

IS SOURCE MOVING TOWARD (-) OR AWAY (+)
FROM US:

A source of a spectral line here might be thought of as a sample of hydrogen. If it is moving towards you a line in its spectrum won't appear at the usual wavelength. It will be at a smaller wavelength. If the hydrogen is moving away from you, the line appears at a larger wavelength than if the hydrogen were standing still relative to you, an effect often called the "red shift."

Once you hit + or - (no RETURN needed) the computer asks for the speed of the source in meters per second. Type this, then hit RETURN. The computer now draws a second spectrum below the first with the line in its Doppler shifted position. Below the two spectra is printed the wavelength of the Doppler shifted line, the unshifted wavelength (observed if the speed of the source is zero relative to you), and the speed you input. The bigger the speed, the larger the shift. You won't see much of a change unless the speed is several million meters per second.

Measurements of wavelengths of spectral lines from objects in space allow calculation of Doppler shifts from wavelengths observed from sources at rest. This in turn allows calculation of the object's speed. Calculations like this have revealed the incredible fact that the entire universe seems to be expanding.

Sometimes your input values of n or strong Doppler shifts produce lines at wavelengths shorter or longer than can be displayed on the spectrum drawn. In this case, the wavelength printed below the spectrum is printed black-on-white instead of white-on-black, and no line will appear on the spectrum.

T H E  E N D

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

ISBN 0-13-938259-3